

Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра програмування та захисту інформації

Програмування на мові Python

Методичні вказівки

до виконання лабораторних робіт студентами
денної та заочної форми навчання спеціальностей
123 "Комп'ютерна інженерія" та
125 "Кібербезпека"

Кропивницький 2017

Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра програмування та захисту інформації

Мелешко Є.В.

Програмування на мові Python

Методичні вказівки

до виконання лабораторних робіт студентами
денної та заочної форми навчання спеціальностей
123 "Комп'ютерна інженерія" та
125 "Кібербезпека"

Затверджено
на засіданні кафедри
програмування та
захисту інформації
Протокол №18
від 26.04.2017

Кропивницький 2017

Програмування на мові Python. Методичні вказівки до виконання лабораторних робіт студентами денної та заочної форми навчання спеціальностей 123 "Комп'ютерна інженерія", 125 "Кібербезпека" / Укл.: Є.В. Мелешко – Кропивницький: ЦНТУ, 2017. – 58 с.

Дані методичні вказівки адресовані майбутнім розробникам програмного забезпечення та фахівцям з аналізу даних, їх рекомендовано використовувати у дисципліні "Програмування" спеціальності 123 "Комп'ютерна інженерія" та дисципліні "Технології програмування" спеціальності 125 "Кібербезпека", також можуть використовуватися студентами інших спеціальностей. Вони включають в себе основи програмування на мові Python, містять основні теоретичні положення та практичні завдання, необхідні для засвоєння навчального матеріалу.

*Схвалено на засіданні методичного семінару
кафедри програмування та захисту інформації
(Протокол № 6 від 04.04.2017)*

Укладач: Мелешко Є.В., канд. техн. наук, доцент

Рецензенти: Смірнов О.А., доктор техн. наук, професор
Минайленко Р.М., канд. техн. наук, доцент

ЗМІСТ

Лабораторна робота № 1. Арифметичні вирази, управляючі конструкції та масиви у мові Python	3
Лабораторна робота № 2. Функції у мові Python.....	19
Лабораторна робота № 3. Робота з файлами у мові Python.....	26
Лабораторна робота № 4. Робота з рядками у мові Python.....	29
Лабораторна робота № 5. Об'єктно-орієнтоване програмування у мові Python.....	39
Лабораторна робота № 6. Збір даних з веб-документів за допомогою мови Python	42
Лабораторна робота № 7. Побудова графіків математичних функцій у мові Python	47
Список літератури	58

Лабораторна робота №1

Тема: Арифметичні вирази, управляючі конструкції та масиви у мові Python

Мета: навчитися створювати найпростіші програми на мові Python, використовуючи оператори вибору і циклів, арифметичні вирази та масиви

Теоретична частина

Python – високорівнева мова програмування загального призначення, орієнтована на підвищення продуктивності розробника і читання коду.

Python підтримує декілька парадигм програмування, в тому числі структурну, об'єктно-орієнтовану, функціональну, імперативну та аспектно-орієнтовану.

На даний час існує дві гілки Python – 2.x та 3.x. Ми будемо працювати переважно з гілкою 3.x.

Існують різні інтерпретатори для мови Python. Офіційний інтерпретатор можна завантажити на сайті <https://www.python.org>

Також Ви можете використати один з варіантів численних збірок Python, наприклад: Anaconda, Python(x, y), EnthoughtCanopy, WinPython тощо.

Основні принципи синтаксису мови Python:

1. Кінець рядка є кінцем інструкції (крапка з комою не потрібна).

```
x = 5
```

```
print(2 + x)
```

2. Вкладені інструкції об'єднуються у блоки за величиною відступів. Відступ може бути будь-яким, головне, щоб в межах одного вкладеного блоку відступ був однаковий (рекомендується робити відступ 4 пробіли).

```
if x == 10:
```

```
    print('yes')
```

3. Вкладені інструкції в Python записуються відповідно до одного і того ж шаблону, коли основна інструкція завершується двокрапкою, слідом за чим розташовується вкладений блок коду, зазвичай з відступом під рядком основної інструкції.

Розглянемо на прикладах основи роботи з Python.

#Базові типи даних (вказуються неявно)

```
a = 5          # int
```

```
b = 7.0        # float
```

```
c = 2>4        # boolean
```

```
d = "World"    # string
```

```
e = 1.5 + 0.5j  # complex
```

```
print(type(a))
```

```
print(type(b))
```

```
print(type(c))
```

```
print(type(d))
```

```
print(type(e))
```

```
print(a, b, c, d, e.real, e.imag)
```

#Перетворення типів даних

```
int(True)  # == 1
float(True) # == 1.0
str(True)  # = 'True'
bool(0)     # == False
bool(0.0)   # == False
bool(1)     # == True
bool(10)    # == True
```

В Python є особливий спосіб обміну змінних значеннями:

```
(a, b) = (b, a)
```

Він використовується дуже часто. Даний метод працює завжди, навіть якщо змінні різних типів (в цьому випадку вони обмінюються не тільки значеннями, але і типами). Круглі дужки в цьому записі можна опустити:

```
a, b = b, a
```

#Основні арифметичні операції

```
a = 5          # int
b = 7.0        # float
c = 1 + 2      # 3
d = 5 - 3      # 2
e = a * b      # 35.0
f = 3.0 / 2    # 1.5
g = 3 / 2      # 1
h = 5 % 3      # 2
j = 10**7.3    # 19952623.1497
```

```
print(type(a))
print(type(b))
print(type(c))
print(type(d))
print(type(e))
print(type(f))
print(type(g))
print(type(h))
print(type(j))
```

```
print(a, b, c, d, e, f, g, h, j)
```

#Вбудовані математичні функції (необхідне підключення бібліотеки math)

```
from math import *
```

```
a = 1
b = 2
```

```
x = sqrt(a*b)/(exp(a)*b)+a*exp((2*a)/b)
```

```
print(x)
```

Таблиця 1 – Функції в бібліотеці math

Назва функції	Призначення функції
math.ceil(x)	Повертає округлене x як найближче ціле значення типу float, яке дорівнює або перевищує x (округлення "вгору").
math.copysign(x, y)	Повертає число x зі знаком числа y. На платформі, яка підтримує знак нуля copysign (1.0, -0.0) дасть -1.0.
math.fabs(x)	Повертає абсолютне значення (модуль) числа x. В Python є вбудована функція abs, але вона повертає модуль числа з тим же типом, що число, fabs же завжди повертає значення типу float.
math.factorial(x)	Повертає факторіал цілого числа x, якщо x не є цілим виникає помилка ValueError.
math.floor(x)	На противагу ceil(x) повертає округлене x як найближче ціле значення типу float, менше або рівне x (округлення "вниз").
math.fmod(x, y)	Аналогічна функції fmod(x, y) бібліотеки C. Зазначимо, що це не те ж саме, що вираз Python $x\%y$. Бажано використовувати при роботі з об'єктами float, в той час як $x\%y$ більше підходить для int.
math.frexp(x)	Являє число в експоненційному записі $x = m \cdot 2^e$ і повертає мантису m (дійсне число, модуль якого лежить в інтервалі від 0.5 до 1) і порядок e (ціле число) як пару чисел (m, e). Якщо x = 0, то повертає (0.0, 0)
math.fsum(iterable)	Повертає float суму від числових елементів ітеруемого об'єкта.
math.isinf(x)	Перевіряє, чи є float об'єкт x плюс або мінус нескінченністю, результат відповідно True або False.
math.isnan(x)	Перевіряє, чи є float об'єкт x об'єктом NaN (not a number).
math.ldexp(x, i)	Повертає значення $x \cdot 2^i$, тобто здійснює дію, зворотну функції math.frexp(x).

math.modf(x)	Повертає частину, що йде після коми і цілу частину від float числа. Обидва результати зберігають знак початкового числа x і представлені типом float.
math.trunc(x)	Повертає цілу частину числа x у вигляді int об'єкта.
Статечні і логарифмічні функції	
math.exp(x)	Повертає exp.
math.log(x [, base])	При передачі функції одного аргументу x, повертає натуральний логарифм x. При передачі двох аргументів, другий береться як основа логарифма.
math.log1p(x)	Повертає натуральний логарифм від $x + 1$.
math.log10(x)	Повертає десятковий логарифм x.
math.pow (x, y)	Повертає x в степені y.
math.sqrt(x)	Квадратний корінь (square root) з x.
Тригонометричні функції	
math.acos(x)	Повертає арккосинус x, в радіанах.
math.asin(x)	Повертає арксинус x, в радіанах.
math.atan(x)	Повертає арктангенс x, в радіанах.
math.atan2(y, x)	Повертає $\text{atan}(y / x)$, в радіанах. Результат лежить в інтервалі $[-\pi, \pi]$. Вектор, кінець, якого задається точкою (x, y) утворює кут з додатнім напрямком осі x. Тому ця функція має більш загальне призначення, ніж попередня. Наприклад й $\text{atan}(1)$, й $\text{atan2}(1, 1)$ дадуть в результаті $\pi/4$, але $\text{atan2}(-1, -1)$ це вже $3\pi/4$.
math.cos(x)	Повертає косинус x, де x виражений в радіанах.
math.hyp(x, y)	Повертає евклідову норму, тобто $\sqrt{x^2+y^2}$. Зручно для обчислення гіпотенузи(hyp) і довжини вектора.
math.sin(x)	Повертає синус x, де x виражений в радіанах.
math.tan(x)	Повертає тангенс x, де x виражений в радіанах.
Радіани в градуси і навпаки	
math.degrees(x)	Конвертує значення кута x з радіан в градуси.
math.radians(x)	Конвертує значення кута x з градусів в радіани.

#Введення даних

x = input('Введіть x\n') #дані, що вводяться мають тип рядка

y = input('Введіть y\n')

x = int(x) #здійснюємо перетворення типів


```
y = int(y)
print (x+y) #додаємо два числа, що були введені користувачем
```

#Виведення даних за форматом

```
for i in range(10):
    A = i*18
    print("%02i\t%.1f" % (i, A))
```

Таблиця 2 – Основні методи форматування даних для виведення

%s	рядок
%d	ціле число
%f	десятькове подання з шістьма знаками після коми
%e	"наукове" подання
%g	компактне подання десяткового числа
%xz	виведення у форматі z в поле ширини x, вирівнювання по правій стороні
%-xz	виведення у форматі z в поле ширини x, вирівнювання по лівій стороні
%.yz	виведення у форматі z з y знаками після коми
%x.yz	виведення у форматі z c y знаками після коми в поле ширини x
%%	виведення знаку процента
\n	перехід на новий рядок

#Оператор умови if

```
for i in range(-50,51,10):
    if i<0:
        print("%i - холодно" % (i))
    elif i>0:
        print("%i - тепло" % (i))
    else:
        print("%i - нормально" % (i))
```

#Цикл for

```
for i in range(10):
    print(i)
```

```
A = []
```

```
for i in range(10):
    A.append(i**i)
```

```
B = zeros(10, dtype=int)
```

```
for i in range(10):  
    B[i] = i**i
```

```
print(A)  
print(B)
```

#Цикл while

```
i = 0  
while i<10:  
    print(i)  
    i = i + 1
```

```
i = 0  
A = []  
while i<10:  
    A.append(i**i)  
    i = i + 1
```

```
print(A)
```

У Python замість масивів використовуються списки, словники, кортежі. Багатомірний масив реалізується за допомогою списку списків (вкладеного списку). За необхідності використовувати саме масиви, напр., при обробці великого об'єму даних, слід застосовувати бібліотеку NumPy, яка дозволяє з ними працювати.

#Списки. Приклад 1

```
a = [1, 2, 3]  
b = [4, 5]  
c = a + b  
d = b * 3  
print(a) # буде виведено [1, 2, 3]  
print(b) # буде виведено [4, 5]  
print(d) # буде виведено [4, 5, 4, 5, 4, 5]  
print([7, 8] + [9]) # буде виведено [7, 8, 9]  
print([0, 1] * 3) # буде виведено [0, 1, 0, 1, 0, 1]
```

```
a2 = [1, 2, 3, 4, 5]  
for i in range(len(a2)):  
    print(a2[i])
```

```
for elem in a:  
    print(elem, end=' ')
```

#Списки. Приклад 2

```
A = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
print(A[0:9:2]) # буде виведено [0, 2, 4, 6, 8]
print(A[0:9:3]) # буде виведено [0, 3, 6]
print(A[2:4]) # буде виведено [2, 3]
print(A[-8:-6]) # буде виведено [2, 3]
print(A[-3:-1]) # буде виведено [7, 8]
```

```
b=list('список')
print(b) # буде виведено ['с', 'п', 'и', 'с', 'о', 'к']
```

#Списки. Приклад 3

```
clouds = ["St", "Sc", "Ns", "As", "Ac"]
```

```
print(type(clouds)) # буде виведено <class 'list'>
print(clouds) # буде виведено ['St', 'Sc', 'Ns', 'As', 'Ac']
```

```
clouds.append("Cu")
print(clouds) # буде виведено ['St', 'Sc', 'Ns', 'As', 'Ac', 'Cu']
```

```
clouds.append("Cb")
print(clouds) # буде виведено ['St', 'Sc', 'Ns', 'As', 'Ac', 'Cu', 'Cb']
```

```
clouds.sort()
print(clouds) # буде виведено ['Ac', 'As', 'Cb', 'Cu', 'Ns', 'Sc', 'St']
```

```
print(clouds[2]) # буде виведено Cb
print(clouds[2:5]) # буде виведено ['Cb', 'Cu', 'Ns']
print(clouds[2:]) # буде виведено ['Cb', 'Cu', 'Ns', 'Sc', 'St']
print(clouds[:5]) # буде виведено ['Ac', 'As', 'Cb', 'Cu', 'Ns']
print(clouds[2:5:2]) # буде виведено ['Cb', 'Ns']
print(clouds[2::2]) # буде виведено ['Cb', 'Ns', 'St']
print(clouds[::-2]) # буде виведено ['Ac', 'Cb', 'Ns', 'St']
```

#Списки списків (можна використати, напр., замість двовимірного масиву)

```
a = [[1, 2, 3], [4, 5, 6]]
print(a) # буде виведено [[1, 2, 3], [4, 5, 6]]
print(a[0]) # буде виведено [1, 2, 3]
print(a[1]) # буде виведено [4, 5, 6]
print(a[0][0]) # буде виведено 1
print(a[0][1]) # буде виведено 2
print(a[1][0]) # буде виведено 4
```

#Списки списків. Приклад створення матриці:

#1 0 0 0

#2 1 0 0

#2 2 1 0

#2 2 2 1

```
n = 4
a = [[0] * n for i in range(n)]
for i in range(n):
    for j in range(n):
        if i < j:
            a[i][j] = 0
        elif i > j:
            a[i][j] = 2
        else:
            a[i][j] = 1
for row in a:
    print(' '.join([str(elem) for elem in row]))
```

Зрізи для списків

example_list[i:j] # вибере всі елементи списку з i-го (включно) по j-й (виключно)

example_list[i:] # вибере всі елементи списку з i-го (включно) до кінця,

example_list[:j] # вибере всі елементи списку з початку по j-й (виключно).

example_list[i:j:k] # вибере кожний k-й елемент списку з i-го (включно) по j-й (виключно)

example_list[i::k] # вибере кожний k-й елемент списку з i-го (включно) до кінця

example_list[:j:k] # вибере кожний k-й елемент списку з початку по j-й (виключно),

example_list[::-k] # вибере кожний k-й елемент списку.

#k також може бути від'ємним, що призведе до формування нового списку із зворотним порядком елементів. Перевірте самі: **example_list[::-2]**, **example_list[:4:3]**, **example_list[1::-1]**, **example_list[3:0:-1]**.

Операції зі списками

x in A – перевірити, чи міститься елемент в списку. Повертає True або False.

x not in A – те ж саме, що not(x in A).

min(A) – найменший елемент списку.

max(A) – найбільший елемент списку.

Методи списків

Метод	Призначення
list.append(x)	Додає елемент в кінець списку
list.extend(L)	Розширює список list, додаючи в кінець всі елементи списку L
list.insert(i, x)	Вставляє на i-ий елемент значення x
list.remove(x)	Видаляє перший елемент у списку, який має значення x. ValueError, якщо такого елемента не існує
list.pop([i])	Видаляє i-ий елемент і повертає його. Якщо індекс не вказано, видаляється останній елемент
list.index(x) list.index(x, [start [, end]])	Повертає положення першого елемента зі значенням x. Якщо вказані параметри start та end – пошук ведеться від start до end
list.count(x)	Повертає кількість елементів зі значенням x
list.sort([key=функція])	Сортує список на основі функції
list.reverse()	Розгортає список у зворотному напрямку
list.copy()	Поверхнева копія списку
list.clear()	Очищає список

Завдання:

В завданні (1) кожного варіанту необхідно обчислити значення z та вивести його на екран.

В завданні (2) кожного варіанту для його реалізації слід застосувати розгалуження та цикли.

В завданні (3) кожного варіанту одномірні масиви слід реалізувати за допомогою списків, а матриці – за допомогою вкладених списків.

Варіанти завдань:

Варіант 1

$$1) z = \cos \alpha + \sin \alpha + \cos 3\alpha + \sin 3\alpha$$

Число α вводиться користувачем у консолі Python.

2) Вивести n -ий елемент послідовності Фібоначчі (у послідовності Фібоначчі кожне наступне число дорівнює сумі двох попередніх: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377. $F_0=1, F_1=1, F_2=1, F_n=F_{n-1}+F_{n-2}, n \geq 2$)

3) Дано матрицю розміром 5×4 . Поміняти місцями перший рядок і третій рядок.

Варіант 2

$$1) z = \frac{1}{4} - \frac{1}{4} \sin\left(\frac{5}{2}\pi - 8\alpha\right)$$

Число α вводиться користувачем у консолі Python.

2) Знайти суму всіх чисел від 1 до n, кратних числу k.

3) Дано одномірний масив, що складається з N дійсних елементів.

- Знайти максимальний елемент.
- Обчислити середнє арифметичне від'ємних елементів масиву.
- Вивести додатні елементи на екран у зворотному порядку.

Варіант 3

$$1) z = \frac{\sqrt{2} - \sqrt{3n}}{2m}$$

Числа m та n вводяться користувачем у консолі Python.

2) Комп'ютер «загадав» число від 1 до 100 (використати функцію random).

Користувач вводять із клавіатури деяке число й одержує одну з відповідей: “Моє число більше”, “Моє число менше”, “Ви вгадали”. Гра повторюється доти поки число не вгадане.

3) Дано одномірний масив, що складається з N цілочисельних елементів.

- Знайти мінімальний елемент.
- Обчислити суму додатних непарних елементів масиву.
- Вивести додатні елементи на екран.

Варіант 4

$$1) z = -\sqrt{m}$$

Число m вводиться користувачем у консолі Python.

2) Знайти добуток всіх непарних чисел від x до y

3) Знайти середнє арифметичне елементів кожного рядка матриці Q(l,m) і відняти його з елементів цього рядка.

Варіант 5

$$1) z = \begin{cases} xy + y, & y < 10 \\ x + y^e, & y \geq 10 \end{cases}$$

Числа x та y вводяться користувачем у консолі Python.

2) Знайти найбільший спільний дільник чисел x та y

3) Дано двовимірний масив розмірністю 4x6, заповнений цілими числами.

Сформувати одномірний масив, кожний елемент якого дорівнює найбільшому елементу відповідного рядка.

Варіант 6

$$1) z = \cos \alpha + \cos 2\alpha + \cos 6\alpha + \cos 7\alpha$$

Число α вводиться користувачем у консолі Python.

2) Дано натуральне число n, обчислити $y = 1 \cdot 3 \cdot 5 \dots (2 \cdot n - 1)$.

3) Дано одномірний масив, що складається з N цілочисельних елементів.

- Знайти максимальний додатний елемент.
- Обчислити суму додатних парних елементів масиву.
- Вивести від'ємні елементи на екран у зворотному порядку.

Варіант 7

$$1) z = \frac{1}{\sqrt{m+2}}$$

Число m вводиться користувачем у консолі Python.

2) Почавши тренування, спортсмен у перший день пробіг 10 км. Щодня він збільшував денну норму на 10% норми попереднього дня. Який сумарний шлях пробіжить спортсмен за n днів?

3) Дано одномірний масив, що складається з N дійсних елементів.

- Знайти мінімальний додатний елемент.
- Обчислити середнє арифметичне додатних елементів масиву.
- Вивести ненульові елементи на екран у зворотному порядку.

Варіант 8

$$1) z = \begin{cases} 3 + y, & x > 8 \\ 9x * y, & x \leq 8 \end{cases}$$

Числа x та y вводяться користувачем у консолі Python.

2) Обчислити факторіал, використовуючи цикл. Число n вводить користувач. $Z = n!$

3) Дано двовимірний масив розмірністю 4×6 , заповнений цілими числами. Сформувати одномірний масив, кожний елемент якого дорівнює сумі елементів відповідного рядка.

Варіант 9

$$1) z = \operatorname{tg} 3\alpha$$

Число α вводиться користувачем у консолі Python.

2) Обчислити $Z = x^n$, використавши цикл, числа x та n вводяться користувачем.

3) Визначити, чи є в двовимірному масиві стовпець, що складається тільки з додатних або нульових елементів.

Варіант 10

$$1) z = \cos^2 \alpha + \cos^4 \alpha$$

Число α вводиться користувачем у консолі Python.

2) Щомісячна стипендія студента становить A грн., а витрати на проживання перевищують стипендію й становлять B грн. на місяць. Ріст цін щомісяця збільшує витрати на 5%. Складіть програму розрахунку суми грошей, яку необхідно одноразово попросити в батьків, щоб можна було прожити навчальний рік (10 місяців), використовуючи тільки ці гроші й стипендію.

3) Дано одномірний масив, що складається з N дійсних елементів.

- Знайти мінімальний додатний елемент.
- Обчислити добуток непарних елементів масиву.
- Вивести від'ємні елементи на екран.

Варіант 11

1) $z = \frac{1}{\sqrt{m} + \sqrt{2}}$

Число m вводиться користувачем у консолі Python.

- 2) Визначити, чи являється n простим числом.
- 3) Дано одномірний масив, що складається з N дійсних елементів.
- Знайти мінімальний додатний елемент.
 - Обчислити суму парних елементів масиву.
 - Вивести масив на екран у зворотному порядку.

Варіант 12

1) $z = \frac{\sqrt{2}}{2} \sin \frac{\alpha}{2}$

Число α вводиться користувачем у консолі Python.

- 2) Одноклітинна амеба ділиться кожні 3 години на 2 клітини. Визначити скільки буде амеб через n годин.
- 3) Дано одномірний масив, що складається з N дійсних елементів.
- Знайти максимальний від'ємний елемент.
 - Обчислити середнє арифметичне непарних елементів масиву.
 - Вивести від'ємні елементи на екран.

Варіант 13

1) $z = (\cos \alpha - \cos \beta)^2 - (\sin \alpha - \sin \beta)^2$

Числа α та β вводиться користувачем у консолі Python.

- 2) Визначити, чи являється число n надлишковим числом. Надлишкове число – додатне ціле число n , сума додатних дільників (відмінних від n) якого перевищує n . Число 48, наприклад, є надлишковим, оскільки $1+2+3+4+6+8+12+16+24=76$, $76 > 48$.
- 3) Дано одномірний масив, що складається з N дійсних елементів.
- Знайти мінімальний від'ємний елемент.
 - Обчислити середнє арифметичне додатних елементів масиву.
 - Вивести додатні елементи на екран.

Варіант 14

1) $z = \sin(\alpha + \beta) \cdot \sin(\alpha - \beta)$

Числа α та β вводиться користувачем у консолі Python.

- 2) Спортсмен пробігає за 1-й день M км, кожний наступний день він збільшує норму пробігу на $K\%$. Визначите через скільки днів норма пробігу може стати більше 50 км.
- 3) Дано одномірний масив, що складається з N цілочисельних елементів.
- Знайти максимальний від'ємний елемент.
 - Обчислити добуток від'ємних елементів масиву.
 - Вивести ненульові елементи на екран у зворотному порядку.

Варіант 15

1) $z = \cos^2 x + \sin^2 y + \frac{1}{4} \sin^2 2x - 1$

Числа x та y вводиться користувачем у консолі Python.

2) Знайти найменше спільне кратне чисел x та y .

3) Дано одномірний масив, що складається з N цілочисельних елементів.

- Знайти мінімальний елемент.
- Обчислити добуток ненульових непарних елементів масиву.
- Вивести масив на екран у зворотному порядку.

Варіант 16

1) $z = \sqrt{\frac{m+3}{m-3}}$

Число m вводиться користувачем у консолі Python.

2) Дано натуральне число n , обчислити $y = 2 \cdot 4 \cdot 6 \dots (2 \cdot n)$.

3) Дано одномірний масив, що складається з N цілочисельних елементів.

- Знайти максимальний від'ємний елемент.
- Обчислити середнє арифметичне парних елементів масиву.
- Вивести ненульові елементи на екран у зворотному порядку.

Варіант 17

1) $z = \begin{cases} \sin 2a + \cos 2b, & a \geq 15 \\ \sqrt{a + b^2}, & a < 15 \end{cases}$

Числа a та b вводяться користувачем у консолі Python.

2) Дано натуральне число n , обчислити $\frac{2}{1} + \frac{3}{2} + \frac{4}{3} + \dots + \frac{n+1}{n}$.

3) Дано одномірний масив, що складається з N дійсних елементів.

- Знайти максимальний елемент.
- Обчислити суму парних елементів масиву.
- Вивести від'ємні елементи на екран у зворотному порядку.

Варіант 18

1) $z = \cos^2 x + \sin^2 y$

Числа x та y вводяться користувачем у консолі Python.

2) Знайти суму S квадратів чисел від 1 до N

3) Дано одномірний масив, що складається з N цілочисельних елементів.

- Знайти максимальний додатний елемент.
- Обчислити добуток елементів масиву.
- Вивести додатні елементи на екран.

Варіант 19

1) $z = \frac{1 - 2\sin^2 x}{1 + \sin^2 x}$

Число x вводиться користувачем у консолі Python.

2) Знайти суму всіх парних чисел від x до y .

3) Дано одномірний масив, що складається з N цілочисельних елементів.

- Знайти мінімальний додатний елемент.
- Обчислити суму додатних елементів масиву, кратних 3.
- Вивести не нульові елементи на екран.

Варіант 20

$$1) z = \frac{\sqrt{m} - \sqrt{n}}{m}$$

Числа m та n вводяться користувачем у консолі Python.

2) Визначити, чи являється число n досконалим. Досконале число – натуральне число, яке дорівнює сумі всіх своїх дільників, напр., 6 (1 + 2 + 3 = 6), 28 (1 + 2 + 4 + 7 + 14 = 28).

3) Дано одномірний масив, що складається з N дійсних елементів.

- Знайти мінімальний елемент.
- Обчислити добуток не нульових елементів масиву.
- Вивести додатні елементи на екран у зворотному порядку.

Варіант 21

$$1) z = \begin{cases} -\sqrt{x}, & x > 45 \\ \sin 2x, & x \leq 45 \end{cases}$$

Число x вводиться користувачем у консолі Python.

2) Знайти перше число Фібоначчі, що буде більше заданого числа p (у послідовності Фібоначчі кожне наступне число дорівнює сумі двох попередніх: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377. F0=1, F1=1, F2=1, Fn=Fn-1+Fn-2, n>=2).

3) Дано одномірний масив, що складається з N дійсних елементів.

- Знайти максимальний елемент.
- Обчислити середнє арифметичне від'ємних елементів масиву.
- Вивести масив на екран у зворотному порядку.

Варіант 22

$$1) z = \frac{\sqrt{2}}{2} \sin \frac{1}{x} + 1$$

Число x вводиться користувачем у консолі Python.

2) Визначити, чи являється число n недостатнім числом. Недостатнє число – натуральне число, сума власних дільників якого менша за саме число. Напр., 15 – недостатнє число, його дільниками є 1, 3 та 5, їх сума рівна 9, що менше 15.

3) Дано одномірний масив, що складається з N цілочисельних елементів.

- Знайти максимальний додатний елемент.
- Обчислити суму елементів масиву.
- Вивести ненульові елементи на екран у зворотному порядку.

Варіант 23

$$1) z = \frac{\sqrt{(3x+2)^2 - 24x}}{3\sqrt{x} - \frac{2}{\sqrt{x}}}$$

Число x вводиться користувачем у консолі Python.

2) Поміняти порядок цифр числа n на зворотній.

3) Дано одномірний масив, що складається з N цілочисельних елементів.

- Знайти максимальний від'ємний елемент.
- Обчислити суму від'ємних елементів масиву.
- Вивести додатні елементи на екран.

Варіант 24

1) $z = 2\sin x + \sqrt{x}$

Число x вводиться користувачем у консолі Python.

2) Дано натуральне число N . Визначити найбільшу цифру і її позицію в числі (напр., $N=573863$, найбільшою є цифра 8, її позиція - четверта зліва).

Для виконання завдання використати операції цілочисельного ділення та знаходження залишку від ділення.

3) Дано одномірний масив, що складається з N цілочисельних елементів.

- Знайти максимальний елемент.
- Обчислити середнє арифметичне непарних елементів масиву.
- Вивести від'ємні елементи на екран.

Варіант 25

1) $z = \cos^e x - \sqrt{x}$

Число x вводиться користувачем у консолі Python.

2) Дано ціле число M . Потрібно знайти найменше ціле від'ємне число k , при якому $3 \nmid k > M$.

3) Дано одномірний масив, що складається з N дійсних елементів.

- Знайти мінімальний від'ємний елемент.
- Обчислити добуток ненульових елементів масиву, кратних 3.
- Вивести від'ємні елементи на екран у зворотному порядку.

Варіант 26

1) $z = \frac{2\operatorname{tg} x - \sqrt{x}}{x}$

Число x вводиться користувачем у консолі Python.

2) Знайти суму всіх чисел від x до y , кратних числу 3.

3) Дано одномірний масив, що складається з N дійсних елементів.

- Знайти мінімальний додатний елемент.
- Обчислити добуток не нульових елементів масиву.
- Вивести ненульові елементи на екран у зворотному порядку.

Варіант 27

1) $z = e^x + \sqrt{x}$

Число x вводиться користувачем у консолі Python.

2) Знайти суму цифр числа n .

3) Дано одномірний масив, що складається з N дійсних елементів.

- Знайти максимальний елемент.
- Обчислити середнє арифметичне додатних елементів масиву.
- Вивести від’ємні елементи на екран у зворотному порядку.

Варіант 28

$$1) z = \frac{(m-1)\sqrt{m} - (n-1)\sqrt{n}}{\sqrt{m^3 + nm + m^2 - n}}$$

Числа m та n вводяться користувачем у консолі Python.

2) В інтервалі від A до B знайти суму та кількість всіх цілих чисел, що діляться на 3 без залишку й не діляться на 9 без залишку.

3) Дано одномірний масив, що складається з N цілочисельних елементів.

- Знайти мінімальний елемент.
- Обчислити суму елементів масиву.
- Вивести додатні елементи на екран.

Варіант 29

$$1) z = \frac{e^{\sqrt{x}}}{\sqrt{1-x}}$$

Число x вводиться користувачем у консолі Python.

2) Дано ціле невід’ємне число N . Якщо N - непарне, то вивести добуток $1 \cdot 3 \cdot \dots \cdot N$; якщо N – парне, то вивести добуток $2 \cdot 4 \cdot \dots \cdot N$.

3) Дано одномірний масив, що складається з N цілочисельних елементів.

- Знайти максимальний елемент масиву.
- Обчислити середнє арифметичне елементів масиву.
- Вивести масив на екран у зворотному порядку.

Варіант 30

$$1) z = e^x + \sin x - \cos 2x$$

Число x вводиться користувачем у консолі Python.

2) Дано натуральне число n . Отримати всі прості дільники цього числа.

3) Дано одномірний масив, що складається з N цілочисельних елементів.

- Знайти мінімальний від’ємний елемент.
- Обчислити суму від’ємних елементів масиву.
- Вивести додатні елементи на екран.

Контрольні питання:

1. Які особливості та переваги мови Python Ви знаєте?
2. Назвіть основні принципи синтаксису мови Python.
3. Як здійснюється введення/виведення даних у мові Python?
4. Який синтаксис циклу for у мові Python?
5. Назвіть принципи роботи зі списками у мові Python.

Лабораторна робота №2

Тема: Функції у мові Python

Мета: навчитися створювати власні функції у мові Python,

Теоретична частина

Функції

Функція в python – об'єкт, який приймає аргументи і повертає значення. Зазвичай функція визначається за допомогою інструкції `def`.

Визначимо найпростішу функцію:

```
def add(x, y):
```

```
    return x + y
```

Інструкція **return** каже, що потрібно повернути значення. У нашому випадку функція повертає суму x і y .

Тепер ми її можемо викликати:

```
>>>
```

```
>>> add(1, 10)
```

```
11
```

```
>>> add('abc', 'def')
```

```
'abcdef'
```

Функція може бути будь-якої складності і повертати будь-які об'єкти (списки, кортежі, і навіть функції).

Функція може і не закінчуватися інструкцією `return`, при цьому функція поверне значення `None`:

```
>>>
```

```
>>> def func():
```

```
...     pass
```

```
...
```

```
>>> print(func())
```

```
None
```

Аргументи функції

Функція може приймати будь-яку кількість аргументів чи не приймати їх зовсім. Також поширені функції з довільним числом аргументів, функції з позиційними і іменованими аргументами, обов'язковими і необов'язковими.

```
>>>
```

```
>>> def func(a, b, c=2): # c - необов'язковий аргумент
```

```
...     return a + b + c
```

```
...
```

```
>>> func(1, 2) # a = 1, b = 2, c = 2 (за замовчуванням)
```

```
5
```

```
>>> func(1, 2, 3) # a = 1, b = 2, c = 3
```

```
6
```

```
>>> func(a=1, b=3) # a = 1, b = 3, c = 2
```

```
6
```

```
>>> func(a=3, c=6) # a = 3, c = 6, b не визначений - виникне помилка
```

Функція також може приймати змінну кількість позиційних аргументів, тоді перед ім'ям ставиться *:

```
>>>
>>> def func(*args):
...     return args
...
>>> func(1, 2, 3, 'abc')
(1, 2, 3, 'abc')
>>> func()
()
>>> func(1)
(1,)
```

Як видно з прикладу, args – це кортеж з усіх переданих аргументів функції, і зі змінною можна працювати так само, як і з кортежем.

Функція може приймати і довільне число іменованих аргументів, тоді перед ім'ям ставиться **:

```
>>>
>>> def func(**kwargs):
...     return kwargs
...
>>> func(a=1, b=2, c=3)
{'a': 1, 'c': 3, 'b': 2}
>>> func()
{}
>>> func(a='python')
{'a': 'python'}
```

В змінній kwargs у нас зберігається словник, з яким ми, знову-таки, можемо робити все, що нам заманеться.

Анонімні функції, інструкція lambda

Анонімні функції можуть містити лише один вислів, але і виконуються вони швидше. Анонімні функції створюються за допомогою інструкції lambda. Крім цього, їх не обов'язково привласнювати змінній, як ми робили з інструкцією def func ():

```
>>>
>>> func = lambda x, y: x + y
>>> func(1, 2)
3
>>> func('a', 'b')
'ab'
>>> (lambda x, y: x + y)(1, 2)
3
>>> (lambda x, y: x + y)('a', 'b')
'ab'
```

lambda функції, на відміну від звичайної, не потрібна інструкція return, а в іншому, вона поводить себе точно так же:

```
>>>
>>> func = lambda *args: args
>>> func(1, 2, 3, 4)
(1, 2, 3, 4)
```

```
f = lambda x: x**2 + 4
Те ж саме, що:
def f(x):
    return x**2 + 4
```

У загальному випадку будь-яка конструкція виду:

```
def g(arg1, arg2, arg3, ...):
    return expression
```

Може бути записана як:

```
g = lambda arg1, arg2, arg3, ...:expression
```

Lambda-функції дуже зручні для того, щоб визначати невеликі функції "на льоту" і тому дуже популярні серед багатьох програмістів.

Lambda-функції часто використовуються для швидкого визначення функції як аргумент іншої функції.

Doc strings

В Python є домовленість про те, що рядки документації (doc strings) вставляються відразу після заголовка функції. Doc strings містять короткий опис мети функції та пояснюють сенс аргументів і значень. Doc strings розміщуються в потрібних лапках `"""`, які дозволяють розбивати текст між ними в кілька рядків. Ось два приклади використання рядків документації у функціях, короткий і довгий:

```
def C2F(C):
    """Convert Celsius degrees (C) to Fahrenheit."""
    return (9.0/5)*C + 32
```

```
def line(x0, y0, x1, y1):
    """
    Compute the coefficients a and b in the mathematical
    expression for a straight line y = a*x + b that goes
    through two points (x0, y0) and (x1, y1).

    x0, y0: a point on the line (floats).
    x1, y1: another point on the line (floats).
    return: coefficients a, b (floats) for the line (y=a*x+b).
    """
    a = (y1 - y0)/float(x1 - x0)
    b = y0 - a*x0
    return a, b
```

Запам'ятайте, що рядки документації повинні розташовуватися до всіх інструкцій функції. Doc strings це не просто коментарі, вони мають більші можливості. Наприклад, для вище описаної функції лінії виконується команда:

```
print(C2F.__doc__)
```

в результаті якої (як ви можете перевірити самі) виводиться весь укладений вміст Doc strings.

Модулі

Підключення модуля зі стандартної бібліотеки

Підключити модуль можна за допомогою інструкції імпорту. Наприклад, підключимо модуль OS для отримання поточної директорії:

```
>>>  
>>> import os  
>>> os.getcwd()  
'C:\\Python33'
```

Після ключового слова `import` вказується назва модуля. Однією інструкцією можна підключити декілька модулів, хоча цього не рекомендується робити, так як це знижує читаність коду. Імпортуємо модулі `time` і `random`.

```
>>>  
>>> import time, random  
>>> time.time()  
1376047104.056417  
>>> random.random()  
0.9874550833306869
```

Після імпортування модуля його назва стає змінною, через яку можна отримати доступ до атрибутів модуля. Наприклад, можна звернутися до константи `e`, розташованої в модулі `math`:

```
>>>  
>>> import math  
>>> math.e  
2.718281828459045
```

Варто відзначити, що якщо зазначений атрибут модуля не буде знайдений, збудиться виняток `AttributeError`. А якщо не вдасться знайти модуль для імпортування, то `ImportError`.

Використання псевдонімів

Якщо назва модуля занадто довга, або вона вам не подобається з якихось інших причин, то для неї можна створити псевдонім, за допомогою ключового слова `as`.

```
>>>  
>>> import math as m  
>>> m.e  
2.718281828459045
```

Тепер доступ до всіх атрибутів модуля `math` здійснюється тільки за допомогою змінної `m`, а змінної `math` в цій програмі вже не буде (якщо,

звичайно, ви після цього не напишете `import math`, тоді модуль буде доступний як під ім'ям `m`, так і під ім'ям `math`).

Інструкція `from`

Підключити певні атрибути модуля можна за допомогою інструкції `from`. Вона має кілька форматів:

`from` <Назва модуля> **`import`** <Атрибут 1> [**`as`** <Псевдонім 1>],
[<Атрибут 2> [**`as`** <Псевдонім 2>] ...]

`from` <Назвамодуля> **`import`** *

Перший формат дозволяє підключити з модуля тільки зазначені вами атрибути. Для довгих імен також можна призначити псевдонім, вказавши його після ключового слова `as`.

```
>>>
>>> from math import e, ceil as c
>>> e
2.718281828459045
>>> c(4.6)
5
```

Імпортовані атрибути можна розмістити на декількох рядках, якщо їх багато, для кращої читання коду:

```
>>>
>>> from math import (sin, cos,
...                   tan, atan)
```

Другий формат інструкції `from` дозволяє підключити всі (точніше, майже все) змінні з модуля. Для прикладу імпортуємо всі атрибути з модуля `sys`:

```
>>>
>>> from sys import *
>>> version
'3.3.2 (v3.3.2:d047928ae3f6, May 16 2013, 00:03:43) [MSC v.1600 32 bit
(Intel)]'
>>> version_info
sys.version_info (major = 3, minor = 3, micro = 2, releaselevel = 'final', serial =
0)
```

Слід зауважити, що не всі атрибути будуть імпортовані. Якщо в модулі визначена змінна `__all__` (список атрибутів, які можуть бути підключені), то будуть підключені тільки атрибути з цього списку. Якщо змінна `__all__` не визначена, то будуть підключені всі атрибути, які не починаються з нижнього підкреслення. Крім того, необхідно враховувати, що імпортування всіх атрибутів з модуля може порушити простір імен головної програми, так як змінні, що мають однакові імена, будуть перезаписані.

Створення свого модуля на Python

Створимо файл `mymodule.py`, в якому визначимо якісь функції:

```
def hello():
    print('Hello, world!')
```

```
def fib(n):
    a = b = 1
    for i in range(n - 2):
        a, b = b, a + b
    return b
```

Тепер в цій же папці створимо інший файл, наприклад, main.py:

```
import mymodule
```

```
mymodule.hello()
print(mymodule.fib(10))
```

На екран буде виведено:

```
Hello, world!
```

```
55
```

Як назвати модуль?

Пам'ятайте, що ви (або інші люди) будуть його імпортувати і використовувати в якості змінної. Модуль неможна назвати так, як і ключове слово. Також імена модулів неможна починати з цифри. І не варто називати модуль так, як будь-яку з вбудованих функцій. Тобто, звичайно, можна, але це створить великі незручності при його подальшому використанні.

Куди помістити модуль?

Туди, де його потім можна буде знайти. Шляхи пошуку модулів вказані в змінній sys.path. У нього включені поточна директорія (тобто модуль можна залишити в папці з основною програмою), а також директорії, в яких встановлено python. Крім того, змінну sys.path можна змінювати вручну, що дозволяє покласти модуль в будь-яке зручний для вас місце (головне, не забути в головній програмі модифікувати sys.path).

Чи можна використовувати модуль як самостійну програму?

Можна. Однак треба пам'ятати, що при імпортуванні модуля його код виконується повністю, тобто, якщо програма щось друкує, то при її імпортуванні це буде надруковано. Цього можна уникнути, якщо перевіряти, чи запущений скрипт як програма, або імпортований. Це можна зробити за допомогою змінної __name__, яка визначена в будь-якій програмі, і дорівнює "__main__", якщо скрипт запущений в якості головної програми, і ім'я, якщо він імпортований. Наприклад, mymodule.py може виглядати ось так:

```
def hello():
    print('Hello, world!')
```

```
def fib(n):
    a = b = 1
    for i in range(n - 2):
        a, b = b, a + b
    return b
```

```
if __name__ == "__main__":  
    hello()  
    for i in range(10):  
        print(fib(i))
```

Завдання:

Розробіть функції для здійснення наступних операцій зі списками:

1. Швидке сортування;
2. Пошук елемента за значенням;
3. Пошук послідовності елементів;
4. Пошук перших п'яти мінімальних елементів;
5. Пошук перших п'яти максимальних елементів;
6. Пошук середнього арифметичного;
7. Повернення списку, що сформований з початкового списку, але не містить повторів (залишається лише перший з однакових елементів).

Помістіть функції в окремий модуль. Реалізуйте програму, яка використовує всі функції зі створеного модуля. Зробити описи Doc strings для кожної реалізованої функції.

Контрольні питання:

1. Як створити функцію у мові Python?
2. Що таке модулі та пакети?
3. Які бувають способи підключення модулів та пакетів?
4. Що таке анонімні функції та інструкція lambda?
5. Чи можна використати модуль як самостійну програму?

Лабораторна робота №3

Тема: Робота з файлами у мові Python

Мета: навчитися здійснювати операції читання та запису для файлів у мові Python

Теоретична частина

Розглянемо на прикладах основні операції з файлами у мові Python

Запис даних у файл

```
filename = "mydata.txt"

# Відкриття файлу
fd = open(filename, "w")

# Запис у файл
for i in range(10):
    A = i*18
    fd.write("%i\t%.1f\n" % (i, A))

# Закриття файлу
fd.close()
```

Читання з файлу

```
import csv
import sys

filename = "mydata.txt"

# Відкриття файлу
fd = open(filename, "r")

# Читання даних
reader = csv.reader(fd, delimiter="\t")

# Виведення змісту файлу
for row in reader:
    print(row)

# Закриття файлу
fd.close()
```

Копіювання файлу

```
import shutil
shutil.copyfile("C:\\mydoc.doc", "C:\\My Documents\\mydoc_2.doc")
```

Перейменування файлу

```
import os
os.rename("C:\\mydoc.doc\\testfile.txt", "/home/user/test.txt")
```

Видалення файлу

```
import os
os.remove(" C:\\mydoc.doc \\testfile.txt")
```

Читання потрібного рядка з текстового файлу

Щоб прочитати рядок під певним номером - можна скористатися як стандартним читанням файлу в список, так і використовувати модуль `linecache`:

```
line = linecache.getline("C:\\boot.ini", 2)
# or
line = open("C:\\boot.ini").readlines()[1]
```

Перебір файлів у каталозі

```
for filename in os.listdir("../plugins"):
    print(filename)
```

Перебір файлів у каталозі за маскою

```
import glob
for filename in glob.glob("../plugins\\*.zip"):
    print(filename)
```

Порівняння файлів

Порівнювати файли можна як за змістом, так і за їх властивостями, що значно швидше. Обидва варіанти можливі за допомогою `filecmp`

```
import filecmp
similar = filecmp.cmp('C:\\file1.txt', 'C:\\file2.txt')
print(similar)
```

Синтаксис функції `Open()` в Python.

```
my_file = open(ім'я_файлу [, режим_доступу][, буферизация])
```

У функції `Open()` багато параметрів, нам поки важливі 3 аргументи: перший, це ім'я файлу. Шлях до файлу може бути відносним або абсолютним. Другий аргумент, це режим, в якому ми будемо відкривати файл.

Режим	Призначення
'r'	відкриття на читання (є значенням за замовчуванням).
'w'	відкриття на запис, вміст файлу видаляється; якщо файлу не існує, створюється новий.
'x'	ексклюзивне створення (збуджується виключення <code>FileExistsError</code> , якщо файл вже існує).
'a'	відкриття на дозапис, інформація додається в кінець файлу.
'b'	відкриття в двійковому режимі.
't'	відкриття в текстовому режимі (є значенням за замовчуванням).
'+'	відкриття на читання і запис

Приклади використання декількох режимів одночасно

Режим	Призначення
r+b	Відкрити бінарний файл на читання і запис. Показчик стоїть на початку файлу.
w+b	Відкрити бінарний файл на читання і запис, очистити до 0 байт. Показчик стоїть на початку файлу.
wb	Відкриває файл для запису в двійковому форматі. Показчик стоїть на початку файлу. Створює файл з ім'ям ім'я_файлу, якщо такого не існує.
a+	Відкриває файл для додавання і читання. Показчик стоїть в кінці файлу. Створює файл з ім'ям ім'я_файлу, якщо такого не існує.

Завдання:

Створіть файли, у яких будуть міститися рядки з іменами студентів та їх середніми балами. Кожен файл буде відповідати окремій групі.

Реалізуйте читання файлів, запис та дозапис у файли, пошук файлів у каталозі та пошук даних у файлі. Також реалізуйте сортування даних у файлі за середнім балом.

Контрольні питання:

1. Як здійснюється запис даних у файл у мові Python?
2. Як здійснюється читання даних з файлу у мові Python?
3. Як здійснюється копіювання файлу?
4. Який синтаксис функції `Open()` у мові Python?
5. Які бувають режими роботи з файлами?

Лабораторна робота №4

Тема: Робота з рядками у мові Python

Мета: набути навичок роботи з вбудованими функціями для роботи з рядками у Python

Теоретична частина

Рядок складається з послідовності символів.

Тип рядка `str`.

```
>>> type('2')
```

```
<class 'str'>
```

У мові Пітон немає окремого символного типу. Символ – це просто рядок довжини 1.

Довжина рядків в Пітоні не обмежена або, строго кажучи, обмежена обсягом виділеної оперативної пам'яті.

Рядок з кількох посліпіль літералів буде неявно конкатенованим навіть при відсутності знаку `+`:

```
s = 'Infinite' 'blue' 'sky'
```

Рядки в потрійних лапках можуть містити кілька рядків тексту. Пробіли від початку рядка входять в текст:

```
s = """Happy birthday to you,  
    My darling Findus!  
    Your hostess."""
```

Можна домогтися того ж ефекту і в одинарних лапках, залишаючи в кінці кожного рядка зворотний слеш (але рекомендується так не робити; після `\` не повинно бути пробілу в кінці рядка):

```
s = "Happy birthday to you,\n    My darling Findus!\n    Your hostess."
```

Базові операції

Конкатенація (додавання)

```
>>>
```

```
>>> S1 = 'spam'
```

```
>>> S2 = 'eggs'
```

```
>>> print(S1 + S2)
```

```
'spameggs'
```

- Дублювання рядка

```
>>>
```

```
>>> print('spam' * 3)
```

```
spamspamspam
```

- Довжина рядка

```
>>>
```

```
>>> len('spam')
```

```
4
```

- Доступ за індексом

```

>>>
>>> S = 'spam'
>>> S[0]
's'
>>> S[2]
'a'
>>> S[-2]
'a'

```

Як видно з прикладу, в Python є можливість доступу по негативного індексу, при цьому відлік йде від кінця рядка.

- Отримання зрізу

Оператор вилучення зрізу: [X: Y]. X - початок зрізу, а Y - закінчення; символ з номером Y в зріз не входить. За умовчанням перший індекс дорівнює 0, а другий - довжині рядка.

```

>>>
>>> s = 'spameggs'
>>> s[3:5]
'me'
>>> s[2:-2]
'ameg'
>>> s[:6]
'spameg'
>>> s[1:]
'pameggs'
>>> s[:]
'spameggs'

```

Крім того, можна задати крок, з яким потрібно витягувати зріз.

```

>>>
>>> s[::-1]
'sggemaps'
>>> s[3:5:-1]
"

>>> s[2::2]
'aeg'

```

Інші функції та методи рядків

При виклику методів необхідно пам'ятати, що рядки в Python відносяться до категорії незмінюваних послідовностей, тобто всі функції і методи можуть лише створювати новий рядок.

```

>>>
>>> s = 'spam'
>>> s[1] = 'b' # Помилка!
Traceback (most recent call last):
  File "", line 1, in
    s[1] = 'b'

```

TypeError: 'str' object does not support item assignment


```
>>> s = s[0] + 'b' + s[2:]
```

```
>>> s
```

```
'sbam'
```

Тому всі строкові методи повертають новий рядок, який потім слід привласнити змінній.

Таблиця "Функції та методи рядків"

Функція або метод	Призначення
S = 'str'; S = "str"; S = '''str'''; S = """str"""	літерали рядків
S = "s\np\ta\nbbb"	екрановані послідовності
S = r"C:\temp\new"	неформатовані рядки (пригнічують екранування)
S = b"byte"	рядок байтів
S1 + S2	конкатенація (додавання рядків)
S1 * 3	повторення рядка
S[i]	звернення за індексом
S[i:j:step]	витяг зрізу
len(S)	довжина рядка
S.find(str, [start],[end])	Пошук підрядка в рядку. Повертає номер першого входження або -1
S.rfind(str, [start],[end])	Пошук підрядка в рядку. Повертає номер останнього входження або -1
S.index(str, [start],[end])	Пошук підрядка в рядку. Повертає номер першого входження або викликає ValueError
S.rindex(str, [start],[end])	Пошук підрядка в рядку. Повертає номер останнього входження або викликає ValueError
S.replace(шаблон, заміна)	Заміна шаблону
S.split(символ)	Розбиття рядка по роздільнику
S.isdigit()	Чи складається рядок з цифр
S.isalpha()	Чи складається рядок з цифр літер
S.isalnum()	Чи складається рядок з цифр або літер
S.islower()	Чи складається рядок із символів в нижньому регістрі
S.isupper()	Чи складається рядок із символів у верхньому регістрі

S.isspace()	Чи складається рядок з невідображаних символів (пробіл, символ переводу сторінки ('\f'), "новий рядок" ('\n'), "перевод каретки" ('\r'), "горизонтальна табуляція" ('\t') і "вертикальна табуляція" ('\v'))
S.istitle()	чи починаються слова в рядку з великої літери
S.upper()	Перетворення рядка до верхнього регістру
S.lower()	Перетворення рядка до нижнього регістру
S.startswith(str)	Чи починається рядок S з шаблону str
S.endswith(str)	Чи закінчується рядок S з шаблону str
S.join(список)	Збірка рядка зі списку з роздільником S
ord(символ)	код символу в ASCII
chr(число)	код ASCII в символ
S.capitalize()	Переводить перший символ рядка в верхній регістр, а всі інші в нижній
S.center(width, [fill])	Повертає відцентрований рядок, по краях якого стоїть символ fill (пробіл за замовчуванням)
S.count(str, [start],[end])	Повертає кількість неперетинаючихся входжень підрядка в діапазоні [початок, кінець] (0 і довжина рядка за замовчуванням)
S.expandtabs([tabsize])	Повертає копію рядка, в якій всі символи табуляції замінюються одним або декількома пробілами, в залежності від поточного стовпця. Якщо TabSize не вказано, розмір табуляції вважається рівним 8 пробілів
S.lstrip([chars])	Видалення символів пробілів на початку рядка
S.rstrip([chars])	Видалення символів пробілів в кінці рядка
S.strip([chars])	Видалення символів пробілів на початку і в кінці рядка
S.partition(шаблон)	Повертає кортеж, що містить частину перед першим шаблоном, сам шаблон, і частина після шаблону. Якщо шаблон не знайдений, повертається кортеж, що містить самий рядок, а потім два порожніх рядка
S.rpartition(sep)	Повертає кортеж, що містить частину перед останнім шаблоном, сам шаблон, і частину після шаблону. Якщо шаблон не знайдений, повертається кортеж, що містить два порожні рядки, а потім сам рядок

S.swapcase()	Переводить символи нижнього регістра в верхній, а верхнього - в нижній
S.title()	Першу букву кожного слова переводить в верхній регістр, а всі інші в нижній
S.zfill(width)	Робить довжину рядка не меншою width, в разі потреби заповнюючи перші символи нулями
S.ljust(width, fillchar=" ")	Робить довжину рядка на меншою width, в разі потреби заповнюючи останні символи символом fillchar
S.rjust(width, fillchar=" ")	Робить довжину рядка не меншою width, в разі потреби заповнюючи перші символи символом fillchar
S.format(*args, **kwargs)	форматування рядка

Екрановані послідовності, так звані escape-послідовності, можуть складатися з одного або декількох символів після зворотної косої риски:

Послідовність	Призначення
\ в самому кінці рядка	ігнорується, рядок продовжується на новому рядку
\\	сам символ зворотного слеша (залишається один символ \)
\'	апостроф (залишається один ')
\"	лапки (залишається один символ ")
\n	новий рядок (новий рядок)
\r	повернення каретки
\t	горизонтальна табуляція
\u...	16-бітовий символ Юнікоду в 16-ковий поданні
\U...	32-бітовий символ Юнікоду в 32-ковий поданні
\x...	16-кове значення
\o...	8-кове значення
\0	Символ Null (не ознака кінця рядка)

Зріз (slice) – вилучення з цього рядка одного символу або деякого фрагмента (підрядка).

Є три форми зрізів. Найпростіша форма зрізу: взяття одного символу рядка, а саме, S[i] – це зріз, що складається з одного символу, який має номер i,

при цьому вважаючи, що нумерація починається з числа 0. Тобто якщо $S = \text{'Hello'}$, то $S[0] == \text{'H'}$, $S[1] == \text{'e'}$, $S[2] == \text{'l'}$, $S[3] == \text{'l'}$, $S[4] == \text{'o'}$.

Номери символів в рядку (а також в інших структурах даних: списках, кортежі) називаються індексом.

Якщо вказати від'ємне значення індексу, то номер буде відраховуватися з кінця, починаючи з номера 1. Тобто

$S[-1] == \text{'o'}$, $S[-2] == \text{'l'}$, $S[-3] == \text{'l'}$, $S[-4] == \text{'e'}$, $S[-5] == \text{'H'}$.

Або у вигляді таблиці:

Рядок S	H	e	l	l	o
Індекс	$S[0]$	$S[1]$	$S[2]$	$S[3]$	$S[4]$
Індекс	$S[-5]$	$S[-4]$	$S[-3]$	$S[-2]$	$S[-1]$

Якщо ж номер символу в зрізі рядка S більше або дорівнює $\text{len}(S)$, або менше, ніж $-\text{len}(S)$, то при зверненні до цього символу рядка відбудеться помилка `IndexError: string index out of range`.

Зріз з двома параметрами: $S[a: b]$ повертає підрядок з $b-a$ символів, починаючи з символу с індексом a , тобто до символу з індексом b , не включаючи його. Наприклад, $S[1: 4] == \text{'ell'}$, те ж саме вийде якщо написати $S[-4: -1]$. Можна використовувати як позитивні, так і негативні індекси в одному зрізі, наприклад, $S[1: -1]$ - це рядок без першого і останнього символу (зріз починається з символу з індексом 1 і закінчуватись індексом -1, не включаючи його).

При використанні такої форми зрізу помилки `IndexError` ніколи не виникає. Наприклад, зріз $S[1: 5]$ поверне рядок `'ello'`, таким же буде результат, якщо зробити другий індекс дуже великим, наприклад, $S[1: 100]$ (якщо в рядку не більше 100 символів).

Якщо опустити другий параметр (але поставити двокрапку), то зріз береться до кінця рядка. Наприклад, щоб видалити з рядка перший символ (його індекс дорівнює 0, тобто взяти зріз, починаючи з символу з індексом 1), то можна взяти зріз $S[1:]$, аналогічно якщо опустити перший параметр, то зріз береться від початку рядка. Тобто видалити з рядка останній символ можна за допомогою зрізу $S[: -1]$. Зріз $S[:]$ збігається з самим рядком S .

Якщо задати зріз з трьома параметрами $S[a: b: d]$, то третій параметр задає крок, як у випадку з функцією `range`, тобто будуть взяті символи з індексами a , $a + d$, $a + 2 * d$ і т.д. Якщо вказати значення третього параметра, рівне 2, в зріз потрапить кожний другий символ, а якщо взяти значення зрізу, рівне -1, то символи будуть йти у зворотному порядку.

Приклади зрізів

Введено рядок:

```
s = input ()
```

Виведемо третій символ цього рядка:

```
print (s[2])
```

Виведемо передостанній символ цього рядка:

```
print (s[-2])
```

Виведемо перші п'ять символів цього рядка:

```
print (s[0: 5])
```

Виведемо весь рядок, крім останніх двох символів:

```
print (s[: - 2])
```

Виведемо всі символи з парними індексами (вважаючи, що індексація починається з 0, тому символи виводяться починаючи з першого):

```
print (s[:: 2])
```

Виведемо всі символи з непарними індексами, тобто починаючи з другого символу рядка:

```
print (s[1 :: 2])
```

Виведемо всі символи у зворотному порядку:

```
print (s[::- 1])
```

Виведемо всі символи рядка через один в зворотному порядку, починаючи з останнього:

```
print (s[::- 2])
```

Мова програмування Python – сучасна мова програмування, тому вона працює виключно з Unicode-символами (це відноситься до версії 3.x).

Код символу можна визначити за допомогою функції `ord`. Ця функція отримує на вхід рядок, який повинен складатися рівно з одного символу. Функція повертає код цього символу. Наприклад, `ord('A')` поверне число 65.

Зворотна функція отримання по числовому коду його номера називається `chr`.

Словники

Словники в Python – неупорядковані колекції довільних об'єктів з доступом по ключу. Їх іноді ще називають асоціативними масивами або хеш-таблицями.

Щоб працювати зі словником, його потрібно створити. Створити його можна кількома способами. По-перше, за допомогою літерала:

```
>>>
>>> d = {}
>>> d
{}
>>> d = {'dict': 1, 'dictionary': 2}
>>> d
{'dict': 1, 'dictionary': 2}
```

По-друге, за допомогою функції **dict**:

```
>>>
>>> d = dict(short='dict', long='dictionary')
>>> d
{'short': 'dict', 'long': 'dictionary'}
>>> d = dict([(1, 1), (2, 4)])
>>> d
{1: 1, 2: 4}
```

По-третє, за допомогою методу **fromkeys**:

```
>>>
>>> d = dict.fromkeys(['a', 'b'])
>>> d
{'a': None, 'b': None}
>>> d = dict.fromkeys(['a', 'b'], 100)
>>> d
{'a': 100, 'b': 100}
```

По-четверте, за допомогою **генераторів словників**, які дуже схожі на генератори списків.

```
>>>
>>> d = {a: a ** 2 for a in range(7)}
>>> d
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36}
```

Тепер спробуємо додати записи в словник і витягти значення ключів:

```
>>>
>>> d = {1: 2, 2: 4, 3: 9}
>>> d[1]
2
>>> d[4] = 4 ** 2
>>> d
{1: 2, 2: 4, 3: 9, 4: 16}
>>> d['1']
Traceback (most recent call last):
  File "", line 1, in
    d['1']
KeyError: '1'
```

Як видно з прикладу, привласнення по новому ключу розширює словник, привласнення за існуючим ключем перезаписує його, а спроба вилучення неіснуючого ключа породжує виключення. Для уникнення виключення є спеціальний метод, або можна перехоплювати виняток.

Що ж можна ще робити зі словниками? Те ж саме, що і з іншими об'єктами: вбудовані функції, ключові слова (наприклад, цикли `for` і `while`), а також спеціальні методи словників.

Методи словників

dict.clear() – очищає словник.

dict.copy() – повертає копію словника.

dict.fromkeys (seq [, value]) – створює словник з ключами з `seq` і значенням `value` (за замовчуванням `None`).

dict.get(key [, default]) – повертає значення ключа, але якщо його немає, не викидає виняток, а повертає `default` (за замовчуванням `None`).

dict.items() – повертає пари (ключ, значення).

dict.keys() – повертає ключі в словнику.

dict.pop(key [, default]) – видаляє ключ і повертає значення. Якщо ключа немає, повертає default (за замовчуванням кидає виняток).

dict.popitem() – видаляє і повертає пару (ключ, значення). Якщо словник порожній, кидає виняток `KeyError`. Пам'ятайте, що словники не впорядковані.

dict.setdefault(key [, default]) – повертає значення ключа, але якщо його немає, не кидає виняток, а створює ключ із значенням default (за замовчуванням `None`).

dict.update([other]) – оновлює словник, додаючи пари (ключ, значення) з other. Існуючі ключі перезаписуються. Повертає `None` (не новий словник!).

dict.values() – повертає значення в словнику.

Завдання 1:

Створіть програму, яка буде складати випадкові фрази на основі трьох списків зі словами. З кожного списку вона повинна брати випадковим чином слова і поєднувати їх в одну фразу.

Завдання 2:

Візьміть текстовий файл, що містить Вашу улюблену художню книгу.

1. Визначте загальну кількість символів у тексті з пробілами та без пробілів.

2. Визначте загальну кількість слів у тексті, загальну кількість різних слів (без повторів) та кількість унікальних слів, що зустрічаються тільки один раз.

Завдання 3:

Виконайте наступне завдання відповідно до свого варіанту.

Варіант 1. Знайдіть у тексті найдовшу послідовність слів, що повторюється більше одного разу.

Варіант 2. Порівняйте два тексти на наявність однакових послідовностей, що містять не менше 5 слів. (Знадобиться ще один текст, напр., того ж автора).

Варіант 3. Знайдіть у тексті послідовності, що містять не менше 3 слів та повторюються не менше 5 раз, створіть список таких послідовностей та підрахуйте кількість їх повторів.

Варіант 4. Підрахуйте загальну кількість речень у тексті, кількість окличних речень, кількість питальних речень, кількість речень, що закінчуються трикрапкою.

Варіант 5. Складіть списки слів, що характеризують різні емоції - не менше 3 емоцій та не менше 10 слів у кожному списку. Підрахуйте частоту появи у тексті слів з кожного списку. Зробіть висновок про те, які емоції з досліджуваних переважають у тексті.

Варіант 6. Визначте максимальну, мінімальну та середню довжину слів, речень та абзаців у тексті.

Варіант 7. Визначте частоту появи питальних речень у тексті та частоту появи слів в цих реченнях.

Варіант 8. Визначте, які слова найчастіше зустрічаються поряд (перед або після) з вказаним користувачем словом.

Варіант 9. Визначте, 10 найчастіше зустрічаємих послідовностей з N слів у тексті. N вказується користувачем.

Варіант 10. Визначте, з якого слова найчастіше починаються речення у тексті, а також яким найчастіше закінчуються.

Варіант 11. Визначте частоту появи слів з різною довжиною.

Варіант 12. Визначте частоту появи речень з різною кількістю слів.

Варіант 13. Складіть список коренів слів, що характеризують колір; визначте кількість слів у тексті, що характеризують колір, а також те, який колір серед них переважає.

Варіант 14. Визначте процент води у тексті (це кількість "стоп-слів" поділена на загальну кількість слів). Стоп-слова – це слова, які ігноруються при індексації сторінок пошуковими системами, не несуть смислового навантаження, замінюються маркерами і негативно впливають на якість текстів, знижуючи їх корисність. Списки стоп-слів див. в Інтернеті.

Варіант 15. Складіть список слів, що характеризують романтичні почуття (не менше 30 слів у списку) та визначте кількість появи даних слів у тексті і виведіть 5 перших абзаців, де таких слів найбільше.

Контрольні питання:

1. Які базові операції роботи з рядками є у мові Python?
2. Як можна одержувати зрізи рядків? Якими бувають зрізи? Наведіть приклади.
3. Які методи роботи з рядками є у мові Python?
4. Що таке словники? Як з ними працювати? Для чого вони потрібні?
5. Які методи роботи зі словниками є у мові Python?

Лабораторна робота №5

Тема: Об'єктно-орієнтоване програмування у мові Python

Мета: набути навичок роботи з класами у мові Python

Теоретична частина

Клас – це складний користувацький тип даних, що складається з полів та методів. *Поля класу* – це змінні, оголошені всередині класу для збереження даних. *Методи класу* – це функції, оголошені для обробки полів класу, взаємодії з основною програмою та іншими даними.

Об'єкти – це окремі екземпляри класу, по суті змінні цього типу. Оголосивши клас із полями та методами, ви можете створювати скільки завгодно об'єктів, кожен з яких міститиме оголошений в класі набір полів та методів.

Класи оголошуються з ключовим словом **class**, ім'я класу за стандартами Python записується наступним чином: всі слова разом, кожне слово з великої літери.

Для створення екземпляру клас викликається як функція з круглими дужками, повертаючи новий об'єкт, який ви можете зберегти у змінній.

Для методів класу можна вказувати атрибути.

Атрибут – це змінна, метод – це функція. Відмінності методу від функції в тому, що у нього є перший параметр – **self**.

Класи, як і модулі, приховують внутрішню будову, залишаючи на поверхні лише зовнішній "інтерфейс" для використання.

Це поєднання даних та функцій всередині однієї сутності, разом із прихованням внутрішньої будови, називається *інкапсуляцією* і є головним принципом ООП.

При оголошенні класу в дужках можуть бути записані (одне або декілька) імена вже існуючих класів – це називається наслідуванням. В такому випадку новий клас (який називається дочірнім) успадковує всі поля та методи класів перелічених в дужках (які називаються батьківськими).

Модель класу:

```
class ім'я_класу:  
    інструкція 1  
    ....  
    інструкція N
```

Створення об'єкта класу:

```
об'єкт_класу = ім'я_класу()
```

Приклад 1. Програма з використанням класу. Обчислення середнього бала студента з трьох предметів.

```
import math
```

```
class Student():
```

```

def GPA(self, name, e1, e2, e3) :
    self.name = name
    self.e1 = e1
    self.e2 = e2
    self.e3 = e3
    print(self.name, ' - ', ((self.e1 + self.e2 + self.e3)/3))

s1 = Student()
s2 = Student()
s1.GPA('Dmytro', 5, 3, 4)
s2.GPA('Olena', 5, 4, 5)

def main():

    return 0

if __name__ == '__main__':
    main()

```

Класи збирають в собі набори даних (змінних) разом з наборами функцій, що на них діють. Мета полягає в тому, щоб досягти більш модульного коду за допомогою групування змінних і функцій, в невеликі вузли, що легко модифікувати.

Завдання:

1. Розробити клас "домашня бібліотека". Реалізувати можливість роботи з довільним числом книг, пошуку по книгах за декількома параметрами (за автором, за роком видання, за жанром тощо), додавання книг у бібліотеку, видалення книг з неї, доступу до книги за номером. Написати програму, що буде демонструвати всі розроблені елементи класу.

2. Розробити клас для представлення відомостей про успішність студента. Об'єкт класу має містити поля для збереження імені студента та балів, отриманих ним за виконання лабораторних робіт та індивідуального творчого завдання.

Забезпечити наступні методи класу:

- конструктор, який приймає рядок ім'я_студента та словник, що містить налаштування курсу у наступному форматі:

- 1) максимально можлива кількість балів за задачу індивідуального творчого завдання;
- 2) максимально можлива кількість балів за задачу однієї лабораторної роботи;
- 3) кількість лабораторних робіт в курсі;
- 4) частка балів від максимуму, яку необхідно набрати для отримання екзамену автоматом.

- метод, за допомогою якого вносяться дані про кількість спроб здати лабораторну роботу та оцінка за останню спробу.
- метод, за допомогою якого вносяться дані про кількість спроб здати індивідуальне творче завдання та оцінка за останню спробу.
- метод, який повертає кортеж (tuple), що містить дійсне число (суму балів студента за проходження курсу), та логічне значення True або False в залежності від того, чи достатньо цих балів для отримання оцінки за екзамен автоматом.

3. Розробити клас, який наслідує функціональність стандартного типу str і містить 2 нових методи:

1) метод, який приймає 1 аргумент s та повертає True або False в залежності від того, чи містить рядок повтори послідовностей символів довжиною від 3 символів.

2) метод, який повертає True або False в залежності від того, чи є рядок паліндромом. Регістрами символів нехтувати. Порожній рядок вважати паліндромом.

4. Розробити клас "колода карт", що буде включати закритий масив елементів класу "карта". В карті буде зберігатися масть та номер. При створенні екземпляру класу "колода карт", карти у колоді розташовуються випадковим чином. Забезпечити можливість виведення карти за номером розташування у колоді, виведення всіх карт, перемішування колоди, видачі однієї карти з колоди, видачі 6 карт з колоди. Написати програму, що буде демонструвати всі розроблені елементи класу.

5. Розробити клас "англо-російський словник", забезпечити можливість зберігання декількох варіантів перекладу для кожного слова. Забезпечити можливість виведення всіх варіантів перекладу введеного англійського слова.

6. Створити абстрактний клас "Транспортний засіб". На його основі реалізувати класи "Літак", "Автомобіль" та "Корабель". Класи повинні мати можливість задавати та отримувати координати і параметри засобів пересування (вартість, швидкість, рік випуску тощо) задати за допомогою полів. Для літака повинна бути визначена висота, для літака та корабля – кількість пасажирів, для корабля – порт приписки. Динамічні характеристики задати за допомогою методів.

Контрольні питання:

1. Як створити клас у мові Python?
2. Що таке інкапсуляція?
3. Що таке об'єкт?
4. Що таке атрибут?
5. У чому полягає відмінність методу від функції?

Лабораторна робота №6

Тема: Збір даних з веб-документів за допомогою мови Python

Мета: навчитися одержувати дані з html-сторінок та здійснювати їх аналіз, використовуючи можливості мови Python

Теоретична частина

Збір даних

Для збору даних з Інтернет-мережі будемо використовувати модуль **requests**, який дозволяє отримувати доступ до веб-сторінок. Як приклад будемо використовувати сайт новин Hacker News.

Згадаймо, що є два найпоширеніші способи доступу до веб-сторінок: запит типу GET і запит типу POST (насправді видів http-запитів набагато більше). Запит типу GET – це коли ви передаєте серверу якусь інформацію в адресному рядку. Наприклад, якщо ви перейдете за такою адресою:

<https://translate.google.com.ua/?hl=uk#en/uk/python>,

то цим ви просите сервіс Google Translate перевести слово "python" з англійської на українську мову (параметри запиту вказуються після символу "?"). POST-запит – це коли вам потрібно ввести інформацію в яку-небудь форму, наприклад, ввести логін-пароль, який не відображатиметься в адресному рядку браузера.

Ми поки будемо використовувати тільки GET-запити.

Давайте виконаємо два різних GET-запиту до новинного сайту:

```
>>> import requests
>>> r = requests.get("https://news.ycombinator.com/newest")
>>> r.ok
True
>>> r.status_code
200
>>> r = requests.get("https://news.ycombinator.com/abracadabra")
>>> r.ok
False
>>> r.status_code
404
```

Перший запит був виконаний успішно, про що говорить значення `r.ok` і `r.status_code`. Другий запит був виконаний до неіснуючої сторінки, що призвело до помилки 404 – "Сторінку не знайдено".

Доступ до вмісту сторінки можна отримати за допомогою атрибута `text` (для прикладу виведено 100 перших символів):

```
>>> r.text[:100]
'<Html op = "newest"> <head> <meta name = "referrer" content = "origin">
<meta name = "viewport" content = "width ='
```

Як ви бачите це проста HTML-сторінка, з якої нам потрібно витягти цікаву для нас інформацію, а саме:

- заголовок новини;
- автора новини;
- посилання на новину;
- кількість коментарів;
- кількість "лайків", яку набрала стаття.

Наприклад, в наступній новині:

▲ Show HN: Pydb – a lightweight database with Python syntax queries, using ZeroMQ (github.com)
63 points by asrp 3 hours ago | hide | past | web | 11 comments

Рисунок 1

- заголовок → Show HN: Pydb – a lightweight database with Python syntax queries, using ZeroMQ;
- автор → asrp;
- посилання → <https://github.com>;
- кількість коментарів → 11;
- кількість "лайків" → 63.

Для отримання даних з веб-сторінок є безліч різних модулів. Проблема з HTML в тому, що більшість браузерів поводить себе поборливо, і тому в Інтернеті багато погано-написаних (не по стандарту HTML) HTML-сторінок. Втім, обробка навіть не цілком коректного HTML-коду не така складна, якщо під рукою є відповідні інструменти. Ми будемо користуватися модулем BeautifulSoup 4.

Щоб використовувати BeautifulSoup, потрібно передати функції BeautifulSoup текст веб-сторінки (у вигляді одного рядка). Для уникнення появи попереджень, також слід вказувати назву парсеру (тієї програми, яка здійснює обробку HTML) – з метою сумісності можна використовувати `html.parser` (він входить в поставку Python і не вимагає установки), але можна також спробувати використовувати `html5lib`, якщо він встановлений.

```
>>> from bs4 import BeautifulSoup
>>> page = BeautifulSoup(r.text, 'html.parser')
>>> page
<html op="newest"><head><meta content="origin" name="referrer"><meta
content="width=device-width, initial-scal
e=1.0" name="viewport"><link href="news.css?5kjS59ufyw5qyqpjcavc"
rel="stylesheet" type="text/css">
<link href="favicon.ico" rel="shortcut icon">
...
```

Змінна `page` представляє собою не просто вміст HTML-сторінки, це об'єкт, який дозволяє виконувати запити. Наприклад, ми можемо звернутися до тегу `head`, а всередині нього до тегу `title`:

```
>>> page.head.title
<title>New Links | Hacker News</title>
>>> page.head.title.text
```

'New Links | Hacker News'

Для того, щоб краще зрозуміти структуру HTML-сторінки слід скористатися веб-інспектором, який є в більшості сучасних браузерів та переглянути код сторінки.

Якщо ви подивіться на структуру HTML-сторінки, то зможете помітити, що є зовнішня таблиця, яка включає в себе ще три таблиці: заголовок, новинну стрічку (яка в свою чергу також складається з великої кількості рядків) і нижній колонтитул (див. рис. 2).

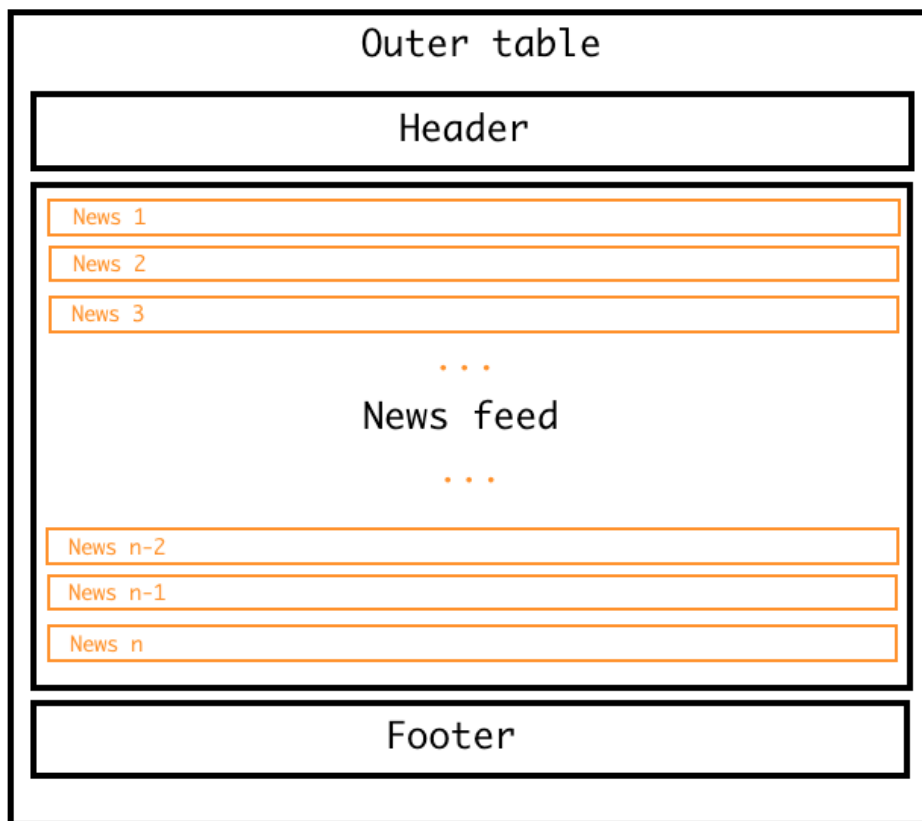


Рисунок 2

Виникає питання як звернутися до внутрішніх таблиць. Якщо ми двічі звернемося до атрибуту table, то отримаємо заголовок:

```
>>> page.table.table
<table border="0" cellpadding="0" cellspacing="0" style="padding:2px"
width="100%"><tr><td style="width:18px; padding-right:4px"><a
href="http://www.ycombinator.com"></a></td>
<td style="line-height:12pt; height:10px;"><span class="pagetop"><b
class="hnname"><a href="news">Hacker News<
/a></b>
<span class="topsel"><a href="newest">new</a></span> | <a
href="newcomments">comments</a> | <a href="show">show<
/a> | <a href="ask">ask</a> | <a href="jobs">jobs</a> | <a
href="submit">submit</a> </span></td><td style="t
```

```

ext-align:right;padding-right:4px;"><span class="pagetop">
<a href="login?goto=newest">login</a>
</span></td>
</tr></table>

```

У об'єкта `page` крім атрибутів є функції, однією з яких є `findAll` і дозволяє знайти декілька елементів з однаковими тегами:

```

>>> tbl_list = page.table.findAll('table')
>>> len(tbl_list)
3

```

Відповідно, нульовий елемент списку `tbl_list` це таблиця, яка містить заголовок, перший елемент списку це таблиця з новинами і другий елемент списку це нижній колонтитул

Збереження даних в `sqlite`

В процесі збору даних їх потрібно десь зберігати. Можна, напр., використовувати для зберігання `SQLite` – компактну вбудовану реляційну базу даних. У стандартній бібліотеці мови `Python` є модуль `sqlite3`, який надає інтерфейс для роботи з `SQLite`. Цей модуль вимагає знання мови `SQL`, тому ми скористаємося іншою технологією, яка називається `ORM`.

`ORM` (англ. `Object-relational mapping`, рус. Об'єктно-реляційне відображення) – технологія програмування, яка зв'язує бази даних з концепціями об'єктно-орієнтованих мов програмування, створюючи "віртуальну об'єктну базу даних". Існують як пропрієтарні, так і вільні реалізації цієї технології.

`SQLAlchemy` – це бібліотека мовою `Python` для роботи з реляційними СУБД з застосуванням технології `ORM`. Служить для синхронізації об'єктів `Python` і записів реляційної бази даних. `SQLAlchemy` дозволяє описувати структури баз даних і способи взаємодії з ними на мові `Python` без використання `SQL`.

Кожна таблиця описується класом, який повинен успадковуватися від базового класу, створеного за допомогою функції `sqlalchemy.ext.declarative.declarative_base()`. У розглянутому нами прикладі буде тільки один клас – `News`, з атрибутами: заголовок, автор, посилання, кількість коментарів і число "лайків".

```

from sqlalchemy.ext.declarative import declarative_base
Base = declarative_base()

```

```

from sqlalchemy import Column, String, Integer
class News(Base):
    __tablename__ = "news"
    id = Column(Integer, primary_key = True)
    title = Column(String)
    author = Column(String)
    url = Column(String)
    comments = Column(Integer)

```

```

points = Column(Integer)
label = Column(String)

from sqlalchemy import create_engine
engine = create_engine("sqlite:///news.db")
Base.metadata.create_all(bind=engine)

from sqlalchemy.orm import sessionmaker
session = sessionmaker(bind=engine)
s = session()

```

Нижче наведено приклад створення об'єкту та збереження його в БД:

```

>>> news = News(title='Lab 7',
                author='dementiy',
                url='https://dementiy.gitbooks.io/-python/content/lab7.html',
                comments=0,
                points=0)

>>> news.id, news.title
(None, Lab 7)
>>> s.add(news)
>>> s.commit()
>>> news.id, news.title
(1, Lab 7)

```

Зверніть увагу, що ідентифікатор об'єкта (id) містить значення None до тих пір, поки ми не зробимо коміт в БД за допомогою методу commit().

Переглянути вміст файлу news.db можна за допомогою програми DB Browser for SQLite.

Завдання:

Реалізуйте програму, яка для довільної сторінки будь-якого сайту новин буде підраховувати частоту появи слів у тексті новини, частоту появи html-тегів, кількість посилань та зображень.

Контрольні питання:

1. Як виконати GET-запит до веб-сайту засобами мови Python?
2. Який модуль/модулі можна використати для збору даних з веб-сторінки?
3. Яку структуру має стандартна HTML-сторінка?
4. Які засоби мова Python надає для роботи з реляційними СУБД?
5. Що таке парсер? Для чого він потрібен?

Лабораторна робота №7

Тема: Побудова графіків математичних функцій у мові Python

Мета: набути навичок роботи з бібліотекою Matplotlib для візуалізації даних

Теоретична частина

Matplotlib – бібліотека на мові програмування Python для візуалізації даних двовимірною 2D графікою (3D графіка також підтримується). Отримувані зображення можуть бути використані як ілюстрації в публікаціях. Зображення, які генеруються в різних форматах, можуть бути використані в інтерактивній графіці, наукових публікаціях, графічному інтерфейсі користувача, веб-додатках, де потрібно будувати діаграми (англ. plotting).

Бібліотека Matplotlib побудована на принципах ООП, але має процедурний інтерфейс pyplot, який надає аналоги команд MATLAB.

Пакет підтримує багато видів графіків і діаграм:

- Графіки (line plot)
- Діаграми розсіювання (scatter plot)
- Стовпчасті діаграми (bar chart) і гістограми (histogram)
- Секторні діаграми (pie chart)
- Діаграми «Стовбур-листя» (stem plot)
- Контурні графіки (contour plot)
- Поля градієнтів (quiver)
- Спектральні діаграми (spectrogram)

Набір підтримуваних форматів зображень, векторних і растрових, можна отримати з словника FigureCanvasBase.filetypes. Типові підтримувані формати: EPS, EMF, JPEG, PDF, PNG, PostScript, RGBA, SVG, SVGZ, TIFF.

Розглянемо побудову графіків на прикладах.

Набір точок

```
>>> import matplotlib.pyplot as plt
>>> plt.plot([1, 3, 2, 4])
[<matplotlib.lines.Line2D object at 0x01A00430>]
>>> plt.show()
```

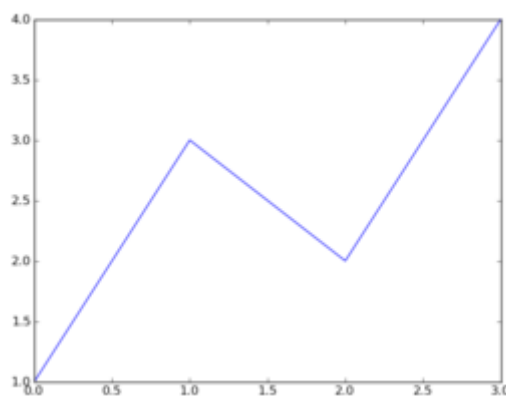


Рисунок 1

Функція `plot()` будує графік, а функція `show()` його показує. Аргумент, що приймається функцією `plot()` – це послідовність у-значень. Інший, який ми опустили, що стоїть перед у – це послідовність х-значень. Оскільки його немає, графік генерується для чотирьох зазначених у, список з чотирьох х: `[0, 1, 2, 3]`.

Функція

```
from numpy import * # для використання функцій експерта linspace  
import matplotlib.pyplot as plt
```

```
def f(t):
```

```
    return t**2*exp(-t**2)
```

```
t = linspace(0, 3, 51) # 51 точка між 0 та 3
```

```
y = f(t)
```

```
plt.plot(t, y)
```

```
plt.show()
```

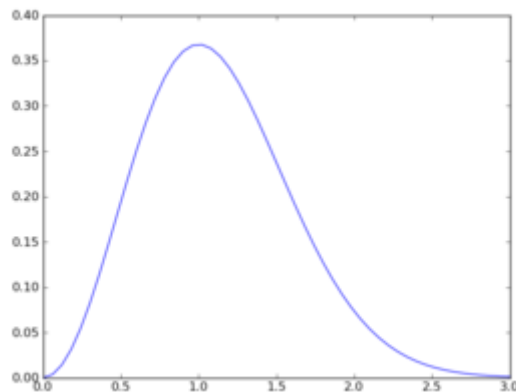


Рисунок 2

Якщо функція більше ніде не використовується, то можна отримати ще більш компактний код, задавши її відразу ж після визначення масиву `t`:

```
from numpy import *  
import matplotlib.pyplot as plt
```

```
t = linspace(0, 3, 51)
```

```
y = t**2*exp(-t**2)
```

```
plt.plot(t, y)
```

```
plt.show()
```

Налаштування вигляду графіків

Крім того, щоб просто побудувати криву, було б добре її назвати, позначити осі, вивести легенду (це особливо стане в нагоді, якщо будувати кілька

графіків). Крім того, інколи потрібно змінити вигляд самої кривої, межі її побудови.

```
from numpy import *
import matplotlib.pyplot as plt

t = linspace(0, 3, 51)
y = t**2*exp(-t**2)

plt.plot(t, y, 'g--', label='t^2*exp(-t^2)')

plt.axis([0, 3, -0.05, 0.5]) # задання [xmin, xmax, ymin, ymax]
plt.xlabel('t') # позначення вісі абсцис
plt.ylabel('y') # позначення е вісі ординат
plt.title('My first normal plot') # назва графіка
plt.legend() # вставка легенди (тексту в label)

plt.show()
```

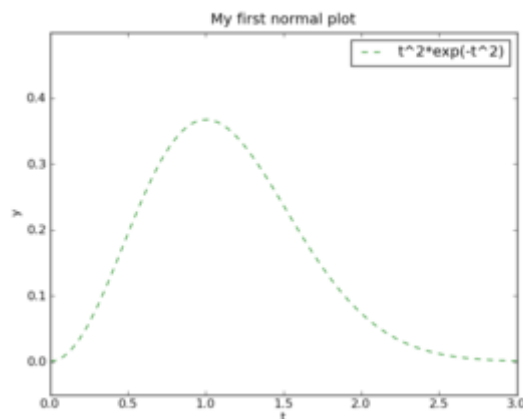


Рисунок 3

Крім зазначених нововведень, позначених в коментарях, в аргументах функції `plot()` ми бачимо два нових. Останній задає текст легенди графіка. Строковий аргумент `g--` відповідальний за те, що змінився вигляд кривої. У порівнянні з попереднім прикладом, графік позеленів (green) і вимальовується - штриховою лінією. За замовчуванням цей аргумент `b-`, що означає синю (blue) суцільну лінію. Нижче наведена таблиця, яка дозволяє вибрати потрібний аргумент.

b, blue	синій колір
c, cyan	блакитний колір
g, green	зелений колір
k, black	чорний колір

m, magenta	пурпурний колір
r, red	червоний колір
w, white	білий колір
y, yellow	жовтий колір
-	суцільна лінія
--	штрихова лінія
-.	штрих-пунктирна лінія
:	пунктирна лінія

Декілька кривих на одному графіку

```

from numpy import *
import matplotlib.pyplot as plt

t = linspace(0, 3, 51)
y1 = t**2*exp(-t**2)
y2 = t**4*exp(-t**2)

plt.plot(t, y1, label='t^2*exp(-t^2)')
plt.plot(t, y2, label='t^4*exp(-t^2)')

# декоративна частина
plt.xlabel('t')
plt.ylabel('y')
plt.title('Plotting two curves in the same plot')
plt.legend()

plt.show()

```

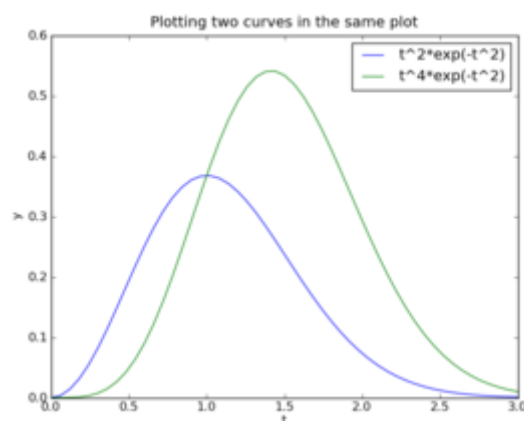


Рисунок 4

Декілька кривих на одному графіку - 2

```

from numpy import *
import matplotlib.pyplot as plt

```

```

t = linspace(0, 3, 51)
y1 = t**2*exp(-t**2)
y2 = t**4*exp(-t**2)
y3 = t**6*exp(-t**2)

plt.plot(t, y1, 'g^', # маркери із зелених трикутників
         t, y2, 'b--', # синя штрихова
         t, y3, 'ro-') # червоні круглі маркери
                        # з'єднані суцільною лінією

plt.xlabel('t')
plt.ylabel('y')
plt.title('Plotting with markers')
plt.legend(['t^2*exp(-t^2)',
           't^4*exp(-t^2)',
           't^6*exp(-t^2)'], # список легенди
          loc='upper left') # положення легенди

plt.show()

```

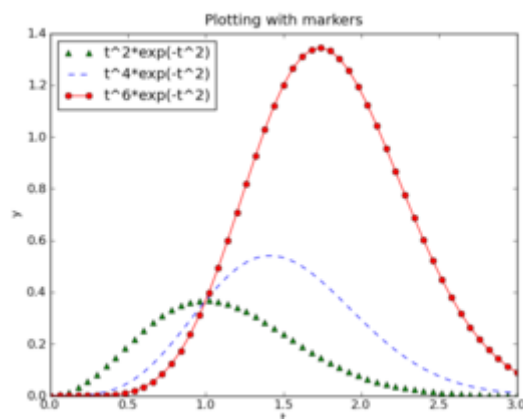


Рисунок 5

Ми змінили положення легенди, інакше б вона закривала червоний графік. За замовчуванням легенда розташовується в правому верхньому куті, але можна її і перенести за рахунок аргументу `loc`. Цьому аргументу можна привласнювати і чисельне значення, але звичайно легше сприймається рядок. У таблиці нижче приводяться можливі варіанти.

Місце	String	Code
кращий варіант	best	0
вгорі справа	upper right	1
вгорі зліва	upper left	2
внизу справа	lower left	3
внизу зліва	lower right	4

справа	right	5
посередині зліва	center left	6
посередині справа	center right	7
посередині внизу	lower center	8
посередині вгорі	upper center	9
посередині	center	10

Маркери

Тут також показано як можна об'єднувати відразу три графіка в одній інструкції. Крім того, видно, що можна не тільки використовувати маркери (y1) або лінії (y2), але і об'єднувати їх разом (y3). Найбільш часто в наукових дослідженнях і журналах призводять графіки, що відрізняються один від одного саме маркерами, тому і в matplotlib для їх позначення є безліч способів:

- . точковий маркер;
- , точки, розміром з піксель;
- o кола;
- ∨ трикутники носом вниз;
- ^ трикутники носом догори;
- > трикутники дивляться вправо;
- < трикутники дивляться вліво;
- s квадрати;
- p п'ятикутники;
- * зірочки;
- h шестикутники;
- H повернені шестикутники;
- + плюси;
- x хрестики;
- D ромби;
- d вузькі ромби;
- | вертикальні зарубки.

Додаткові аргументи plot()

Отже, в один аргумент ми можемо поставити відразу три параметра: першим вказуємо колір, другим – стиль лінії, третім – тип маркера. Однак вже така нотація може у людини незнайомої з нею, викликати подив. Крім того, вона не дозволяє розділяти параметри лінії і маркера, тому існує варіант з використанням keywords – також це дозволяє щедрі функція plot():

Keyword argument	Що міняє
color або c	колір лінії
linestyle	стиль лінії, використовуються позначення, показані вище
linewidth	товщина лінії у вигляді float-числа
marker	вид маркера
markeredgecolor	колір краю (edge) маркера
markeredgewidth	товщина краю маркера

<code>markerfacecolor</code>	колір самого маркера
<code>markersize</code>	розмір маркера

Можна також вносити зміни у відмітки на осях координат. Робиться це за допомогою функцій `xticks()` і `yticks()`, в які передаються один або два списки значень: або просто список згаданих значень, або їх же, але спочатку ті місця, на які вони встають:

```
x = [5, 3, 7, 2, 4, 1]
plt.xticks(range(len(x)), ['a', 'b', 'c', 'd', 'e', 'f'])
plt.yticks(range(1, 8, 2))
```

Для нанесення сітки існує команда:

```
plt.grid(True)
```

Для того, щоб одну або декілька осей виставити в логарифмічному масштабі застосовуються команди `plt.semilogx()` и `plt.semilogy()`.

Збереження файлу

```
from numpy import *
import matplotlib.pyplot as plt
```

```
t = linspace(0, 3, 51)
y = t**2*exp(-t**2)
```

```
plt.plot(t, y)
plt.savefig('name_of_plot.png', dpi=200)
```

Файл зберігається в тій же директорії з ім'ям і розширенням, зазначеним в першому аргументі. Другий необов'язковий аргумент дозволяє «на льоту» змінювати роздільну здатність картинки, що зберігається у файл.

Буває так, що дивитися на картинки в уже налаштованій програмі не потрібно і потрібно їх саме зберігати на майбутнє, щоб переглянути і порівняти їх всі разом. Тоді нам не потрібно запускати вікно перегляду результатів. Для цього до колишніх інструкцій додаємо головному модулю повідомлення:

```
import matplotlib
matplotlib.use('Agg')
```

Гістограми

Для побудови гістограм (діаграм у вигляді набору стовпчиків) в Matplotlib використовуються функція `bar` і `barh`, які будують вертикальні або горизонтальні гістограми відповідно. Ці функції, як і інші функції

малювання, імпортуються з модуля `pylab`. Функції `bar` і `barh` мають безліч необов'язкових параметрів з додатковими настройками, ми розглянемо тільки найбільш часто використовувані можливості для налаштування зовнішнього вигляду гістограм.

Функції `bar` і `barh` мають два обов'язкових параметра:

- Список координат розташування стовпчиків по осі `X` для `bar` або по осі `Y` для `barh`.
- Значення, що задають висоту (довжину) стовпчиків.

Довжини цих двох списків повинні бути рівні.

Найпростіший приклад може виглядати так:

```
import pylab
```

```
xdata = [0, 1, 2, 4, 5, 8]
```

```
ydata = [0.1, 0.2, 0.4, 0.8, 0.6, 0.1]
```

```
pylab.bar(xdata, ydata)
```

```
pylab.show()
```

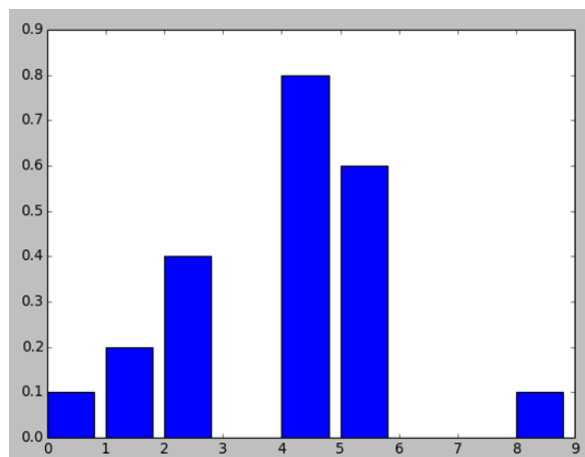


Рисунок 6

```
import matplotlib.pyplot as plt
```

```
import numpy as np
```

```
y = np.random.randn(1000)
```

```
plt.hist(y, 25)
```

```
plt.show()
```

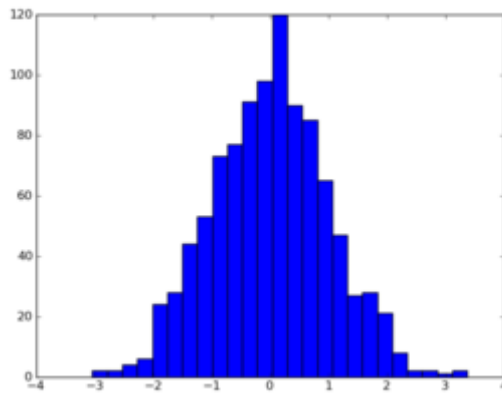



Рисунок 7

```
import matplotlib.pyplot as plt  
import numpy as np
```

```
data1=10*np.random.rand(5)  
data2=10*np.random.rand(5)  
data3=10*np.random.rand(5)
```

```
locs = np.arange(1, len(data1)+1)  
width = 0.27
```

```
plt.bar(locs, data1, width=width)  
plt.bar(locs+width, data2, width=width, color='red')  
plt.bar(locs+2*width, data3, width=width, color='green')
```

```
plt.xticks(locs + width*1.5, locs)  
plt.show()
```

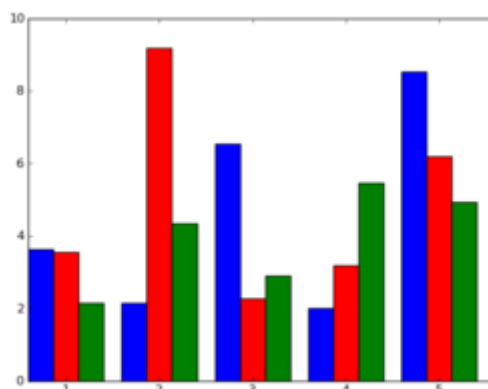


Рисунок 8

Текст, примітки

Крім тексту в назвах, підписах до осей, легенди, можна безпосередньо вставляти його в графік за допомогою простої функції `text(x, y, text)`, де `x` і `y` координати, а `text` текстовий рядок. Ця функція вставляє текст відповідно до

координат даних. Існує можливість вводити і в координатах графіка, в яких за (0, 0) приймається нижній лівий кут, а за (1, 1) правий верхній. Це робиться за допомогою функції `figtext(x, y, text)`.

Текстові функції, звичайно вставляють текст в графік, але часто буває потрібно саме вказати, виділити якийсь екстремум, незвичайну точку. Це легко зробити за допомогою приміток – функції `annotate('annotation', xy = (x1, y1), xytext = (x2, y2))`. Тут замість `annotation` ми пишемо текст примітки, замість `(x1, y1)` координати цікавої точки, замість `(x2, y2)` координати, де ми хочемо вставити текст.

Завдання 1: зобразити 2d графік функції відповідно своєму варіанту та зберегти у .png файл.

Варіанти:

№	Функція
1.	$Y(x)=x*\sin(5*x), x=[-2...5]$
2.	$Y(x)=1/x*\sin(5*x), x=[-5...5]$
3.	$Y(x)=2^x*\sin(10x), x=[-3...3]$
4.	$Y(x)=x^{(1/2)}*\sin(10*x), x=[0...5]$
5.	$Y(x)=15*\sin(10*x)*\cos(3*x), x=[-3...3]$
6.	$Y(x)=5*\sin(10*x)*\sin(3*x), x=[0...4]$
7.	$Y(x)=\sin(10*x)*\sin(3*x)/(x^2), x=[0...4]$
8.	$Y(x)=5*\sin(10*x)*\sin(3*x)/(x^{(1/2)}), x=[1...7]$
9.	$Y(x)=5*\cos(10*x)*\sin(3*x)/(x^{(1/2)}), x=[0...5]$
10.	$Y(x)=-5*\cos(10*x)*\sin(3*x)/(x^{(1/2)}), x=[0...10]$
11.	$Y(x)=-5*\cos(10*x)*\sin(3*x)/(x^x), x=[0...5]$
12.	$Y(x)=5*\sin(10*x)*\sin(3*x)/(x^x), x=[0...8]$
13.	$Y(x)=x^{\sin(10*x)}, x=[1...10]$
14.	$Y(x)=-x^{\cos(5*x)}, x=[0...10]$
15.	$Y(x)=x^{\cos(x^2)}, x=[0...10]$
16.	$Y(x)=\cos(x^2)/x, x=[0...5]$
17.	$Y(x)=10*\cos(x^2)/x^2, x=[0...4]$
18.	$Y(x)=(1/x)*\cos(x^2+1/x), x=[1...10]$
19.	$Y(x)=\sin(x)*(1/x)*\cos(x^2+1/x), x=[-2...2]$
20.	$Y(x)=5*\sin(x)*\cos(x^2+1/x)^2, x=[1...10]$
21.	$Y(x)=5*\sin(1/x)*\cos(x^2+1/x)^2, x=[1...4]$
22.	$Y(x)=5*\sin(1/x)*\cos(x^2)^3, x=[-4...4]$

- 23. $Y(x)=(x^3)*\cos(x^2)$, $x=[-2...2]$
- 24. $Y(x)=(x^3)+\cos(15*x)$, $x=[-2...2]$
- 25. $Y(x)=(3^x)+\cos(15*x)$, $x=[-1...2]$

Завдання 2: Зобразити гістограму частоти появи літер у певному тексті та зберегти у .png файл.

Завдання 3: Зобразити гістограму частоти появи у певному тексті звичайних, питальних та окличних речень, а також речень, що завершуються трикрапкою та зберегти у .png файл.

Контрольні питання:

- 1. Які засоби мова Python надає для роботи з 2D графікою? Які бібліотеки призначені для роботи з графікою?
- 2. Яким чином можна відобразити графік математичної функції?
- 3. Як можна налаштувати колір та тип лінії на графіку математичної функції?
- 4. Яким чином можна відобразити гісторграму?
- 5. Яким чином можна зберегти зображення у файл?

Список літератури

1. Марк Саммерфилд "Программирование на Python 3" . Подробное руководство. – Пер. с англ. – СПб.: Символ-Плюс, 2013. – 608 с.
2. Лутц, М. Программирование на Python, I том / М. Лутц. - СПб.: Символ-плюс, 2015. - 992 с.
3. Лутц, М. Программирование на Python, II том / М. Лутц. - СПб.: Символ-плюс, 2015. - 992 с.
4. МакГрат, М. Программирование на Python для начинающих / М. МакГрат. - М.: Эксмо, 2015. - 192 с.
5. Доусон М. Програмуємо на Python. – СПб.: Питер, 2014. – 416 с.
6. Прохоренок Н.А. Python 3 и PyQt. Разработка приложений. – СПб.: БХВ-Петербург, 2012. – 704 с.
7. Хахаев И.А. Практикум по алгоритмизации и программированию на Python. – М.: Альт Линукс, 2010. — 126 с.
8. Downey A., Elkner J., Meyers Ch. How to Think Like a Computer Scientist: Learning with Python. - Wellesley, Massachusetts: Green Tea Press, 2002. - 290 pp.
9. Tutorialspoint / Python – [Електронний ресурс]. – Режим доступу: <http://www.tutorialspoint.com/python/>
10. Самоучитель Python – [Електронний ресурс]. – Режим доступу: <https://pythonworld.ru/samouchitel-python>
11. Python's documentation, tutorials, and guides are constantly evolving. – [Електронний ресурс]. – Режим доступу: <https://www.python.org/doc/>